FTP 25G Server Demo reference design

Rev1.0 15-Oct-20

1 Introduction

File Transfer Protocol (FTP) is a standard network protocol used for file transmission between a client and server on a network. For the reliable and efficient transmission, TCP/IP is used as lower layer.

Reference documents

1. File Transfer Protocol: http://tools.ietf.org/html/rfc959
2. File Transfer Protocol: http://www.tcpipguide.com/free/t_FileTransferProtocolFTP.htm
3. FTP Sequence: www.eventhelix.com/realtimemantra/networking/FTP.pdf
4. List of FTP commands: http://en.wikipedia.org/wiki/List_of_FTP_commands
5. List of FTP server return codes: http://en.wikipedia.org/wiki/List_of_FTP_server_return_codes

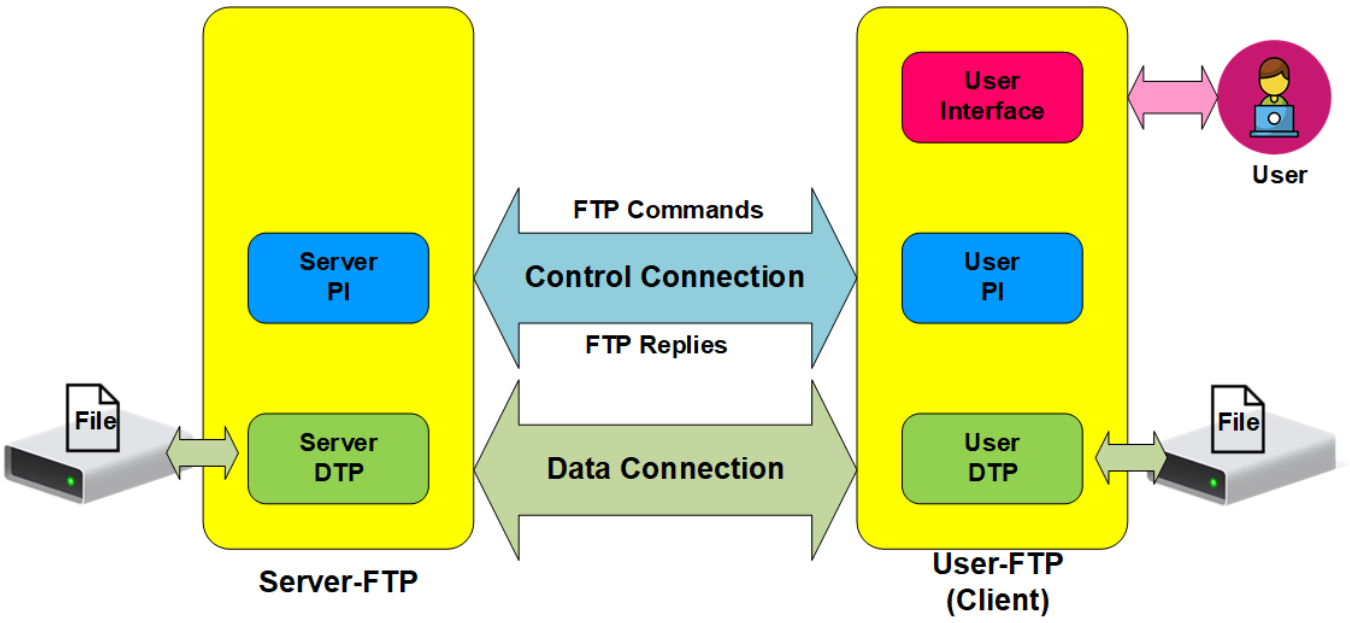

Figure 1-1 Model of FTP Use

In the model described in Figure 1-1, there are two hosts with their own data storage. One is server with shared data in the network and another is client for general users who can authenticate themselves with sign-in protocol, normally in form of username and password. For this protocol, two connections are required. First is control connection for FTP command and FTP replies transferring between client and server. Second is data connection for data transmission between two hosts.

Control connection is taken responsibility by the server-protocol interpreter (Server-PI) and the user-protocol interpreter (User-PI). The server-PI listens on its own port number (Port 21 is used since it is a well-known control port) until the connection is established. After that, Server-PI waits for FTP commands from User-PI, returns standard FTP replies over the control connection in response of the command and manages the server data transfer process (Server-DTP) when the command needs data transfer. For client side, User-PI is responsible to forward the command received from the user interface to Server-PI, wait FTP replies and manage the user data transfer process (User-DTP) when the command needs data transfer.

Data connection is taken responsibility by the server data transfer process (Server-DTP) and the user data transfer process (User-DTP) for sending or receiving data. The data connection establishment can be done by Server-DTP (active mode) or User-DTP (passive mode). Passive mode is commonly used to resolve the firewall problem on FTP client. The data connection is established to transfer information or files between the server and the client. The data connection is terminated after finishing information or file transferring. Both Server-DTP and User-DTP interact with the local file system when reading or writing files.

User Interface provides the interface for a person requiring to obtain file transfer service through the FTP software. The command is issued and the response is checked.

1.1 FTP Connection Establishment and User Authentication

After user creates the connection on control port, the next process is user authentication. FTP server can be accessed by authorized user only. The details of the connection establishment and login process are described as follows.

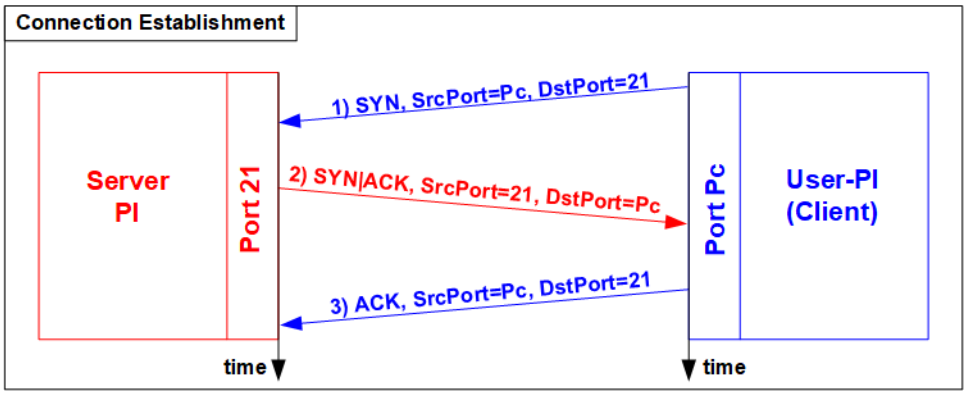

Figure 1-2 Connection Establishment on Control port

According to Transmission Control Protocol (TCP), 3-Way Handshake process is the way to establish the connection, as shown in Figure 1-2. FTP client starts sending TCP packet with SYN flag to server on Port 21 to be the request for creating the new connection. After this packet is detected by FTP Server, the server returns the TCP packet with SYN and ACK flag to allow the new connection creating. Finally, client sends acknowledgement (ACK flag) to finish this process and the control port is in ready status for server and client communication. The control port is applied for transferring FTP commands and replies.

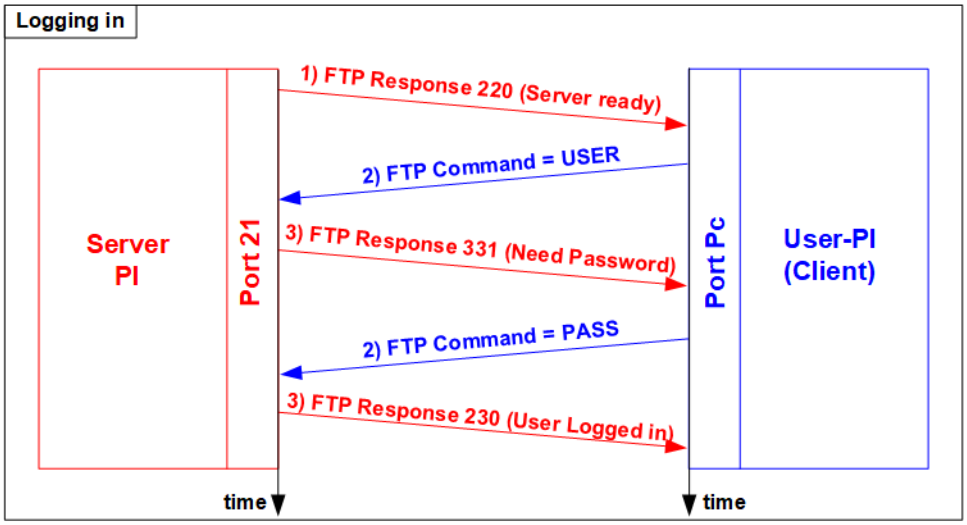

Figure 1-3 User Authentication

As shown in Figure 1-3, after the connection was established, the authorized users have to identify themselves with allowed username and password. For user authentication, client sends two FTP commands which are USER command with username and PASS with password respectively. Finally, the server replies the client with FTP response 230 to allow opening the new session.

1.2 FTP Data Connection Management by Passive Mode

This reference design implements the data connection establishment by Client (Passive mode) which is generally used. So, only the step for running as Passive mode is described in this topic.

Some FTP commands e.g. LIST (List subdirectories or files), STOR (Store files) and RETR (Retrieve files) must transfer information or file through the data connection. Before transferring data in previously mentioned commands, PASV command is firstly sent from client to specify the parameters for the data connection, i.e., IP address and port number of Server-DTP.

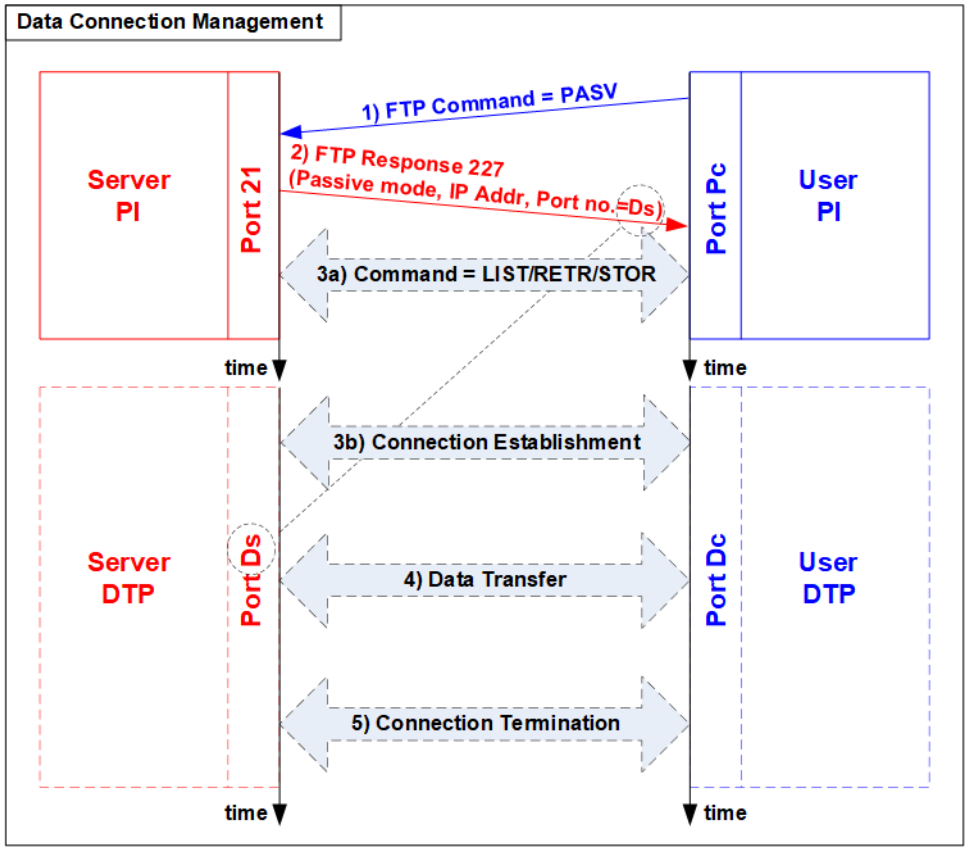

Figure 1-4 Data Connection Management in Passive Mode

After receiving PASV command, the server returns FTP response 227 to inform IP address and port number of Server to Client. Assume the data port number of the server is equal to Ds. Server-PI instructs Server-DTP to listen on Port Ds and wait for data connection establishment. Next, User-PI sends FTP command which requires data transfer to Server-PI. At the same time, User-DTP establishes the data connection to transfer data with Server-DTP. The sequence of Step 3a) for transferring FTP command and step 3b) for the data connection establishment can be swapped, depending on FTP client behavior. When the data transfer is finished, the data connection is terminated by the side transmitting data.

1.3 LIST Command

The LIST command is issued to transfer information about files in the specified directory, stored on the server's side, through an established data connection. PASV command is sent firstly to initialize the data connection. After that, User-PI sends LIST command to Server-PI and User-DTP establishes data connection at the same time. The sequence of Step 2a) for transferring FTP command and step 2b) for the data connection establishment is possibly swapped.

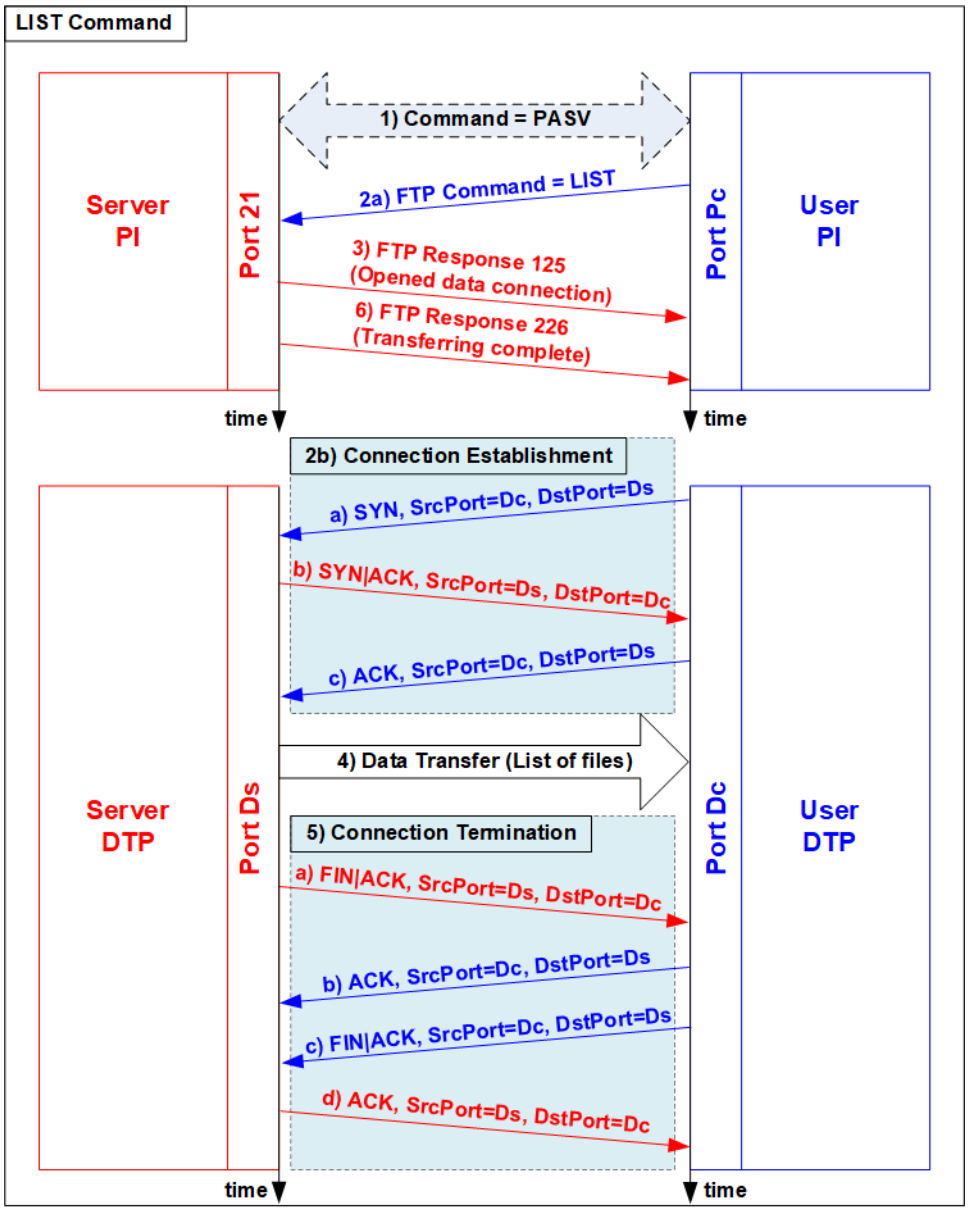

Figure 1-5 LIST Command Operation

For the server side, after the data connection was established, Server-PI returns response 125 for LIST command and Server-DTP returns information of files or the list of files to client. When the data transfer is finished, Server-DTP terminates the data connection. Finally, Server-PI sends response 226 to complete LIST command operation.

1.4 STOR Command

The STOR command is issued by client when the user requires to upload a copy of a file to store on the server's storage. The client provides the file name for uploading with the STOR command. If the file already exists on the server, it is replaced by the uploaded file. Otherwise, the uploaded file is created. Next, the server decodes file name to be a parameter to store a file. After that, data transmission begins, similar to the operation of LIST command. Data transfer direction is opposite from LIST and RETR command by sending from User-DTP (client) to Server-DTP (server). User-DTP terminates the data connection after finishing all data transmission.

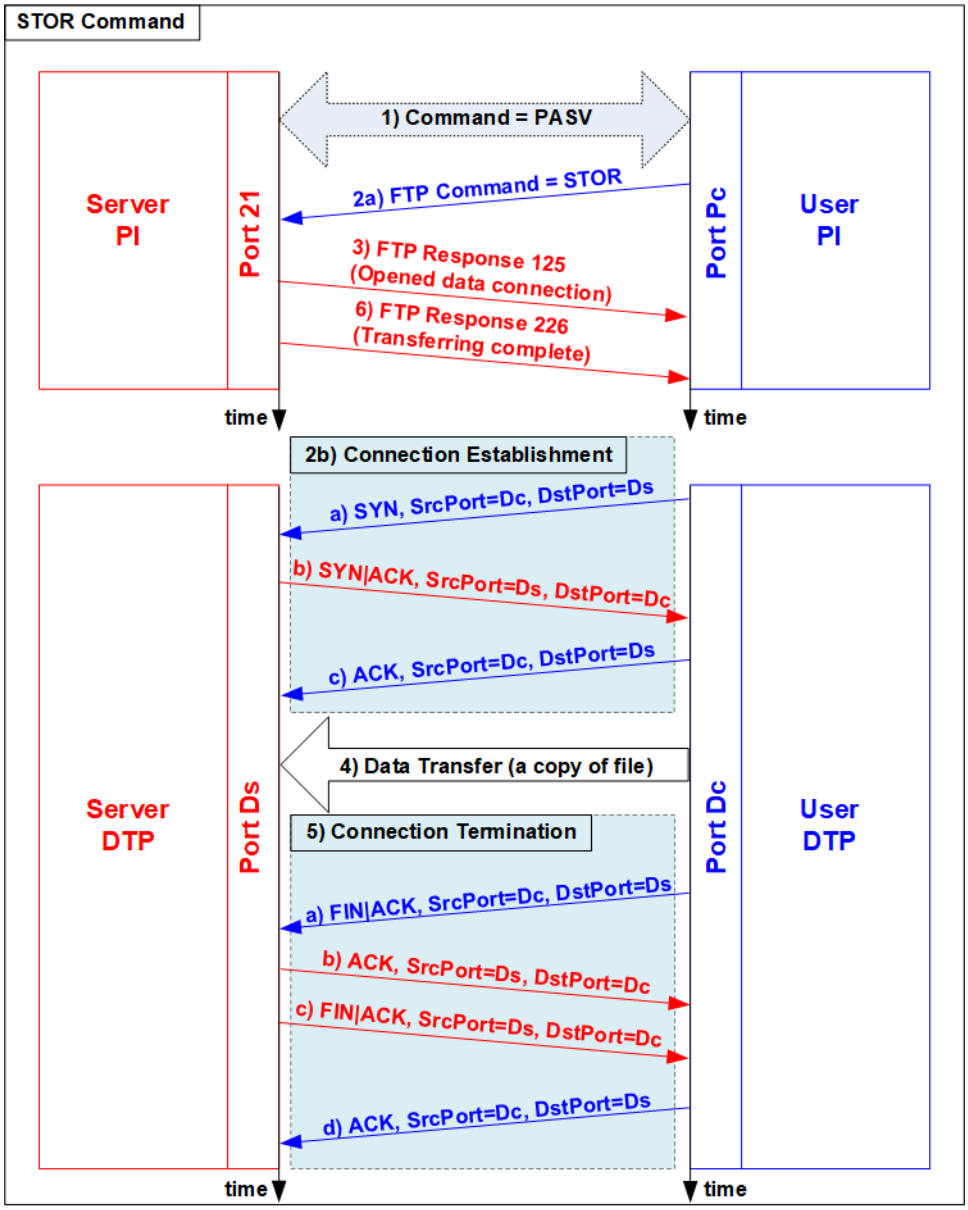

Figure 1-6 STOR Command Operation

1.5 RETR Command

The RETR command is sent from the client when user requires to download a copy of a file on the server. The client provides the file name along with the RETR command. On the server side, after the file name is decoded, the data of the requested file is returned. The step of data sequence is similar to LIST command. The data connection is released by Server-DTP after the data transfer is completed.

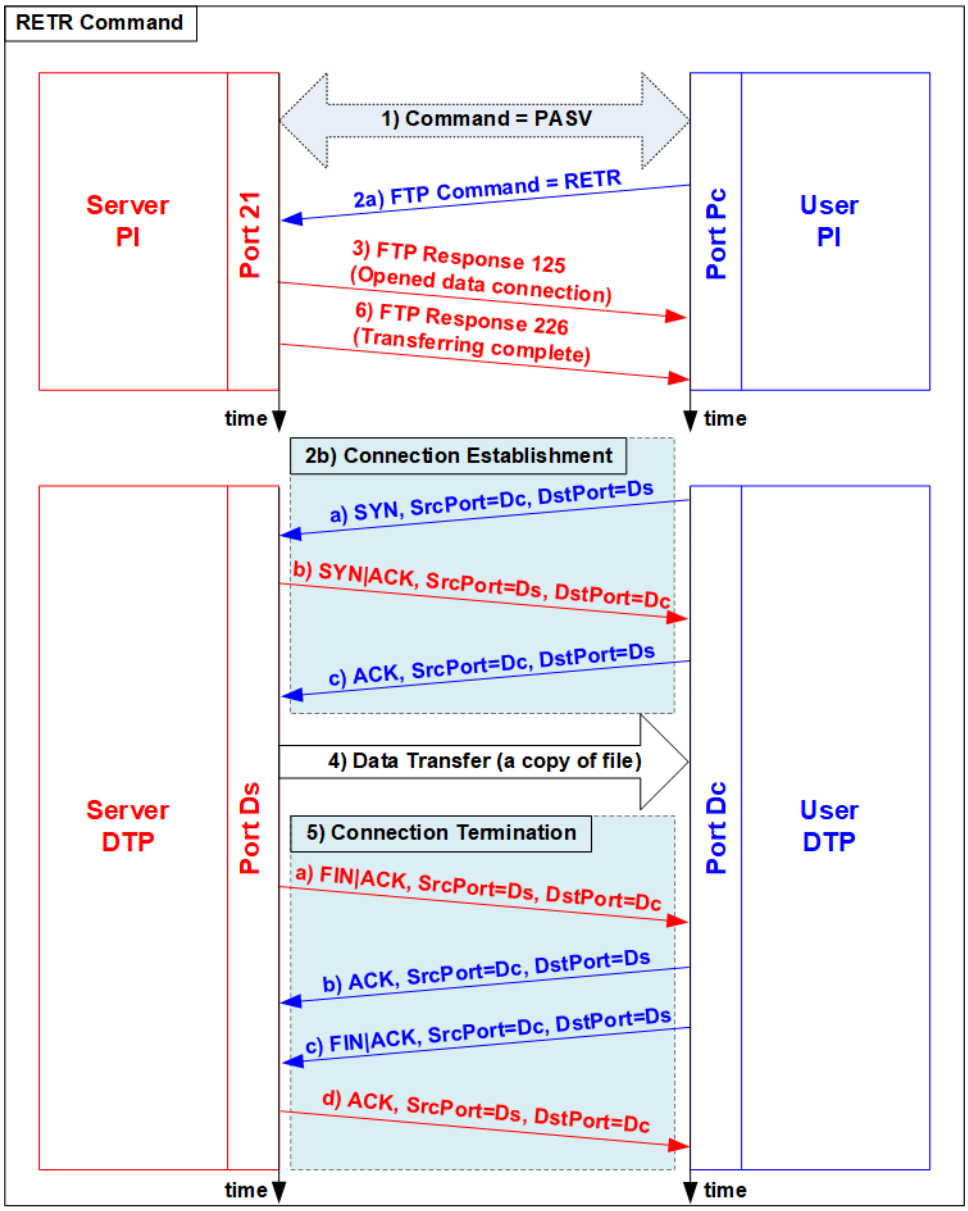

Figure 1-7 RETR Command Operation

2 Hardware Structure

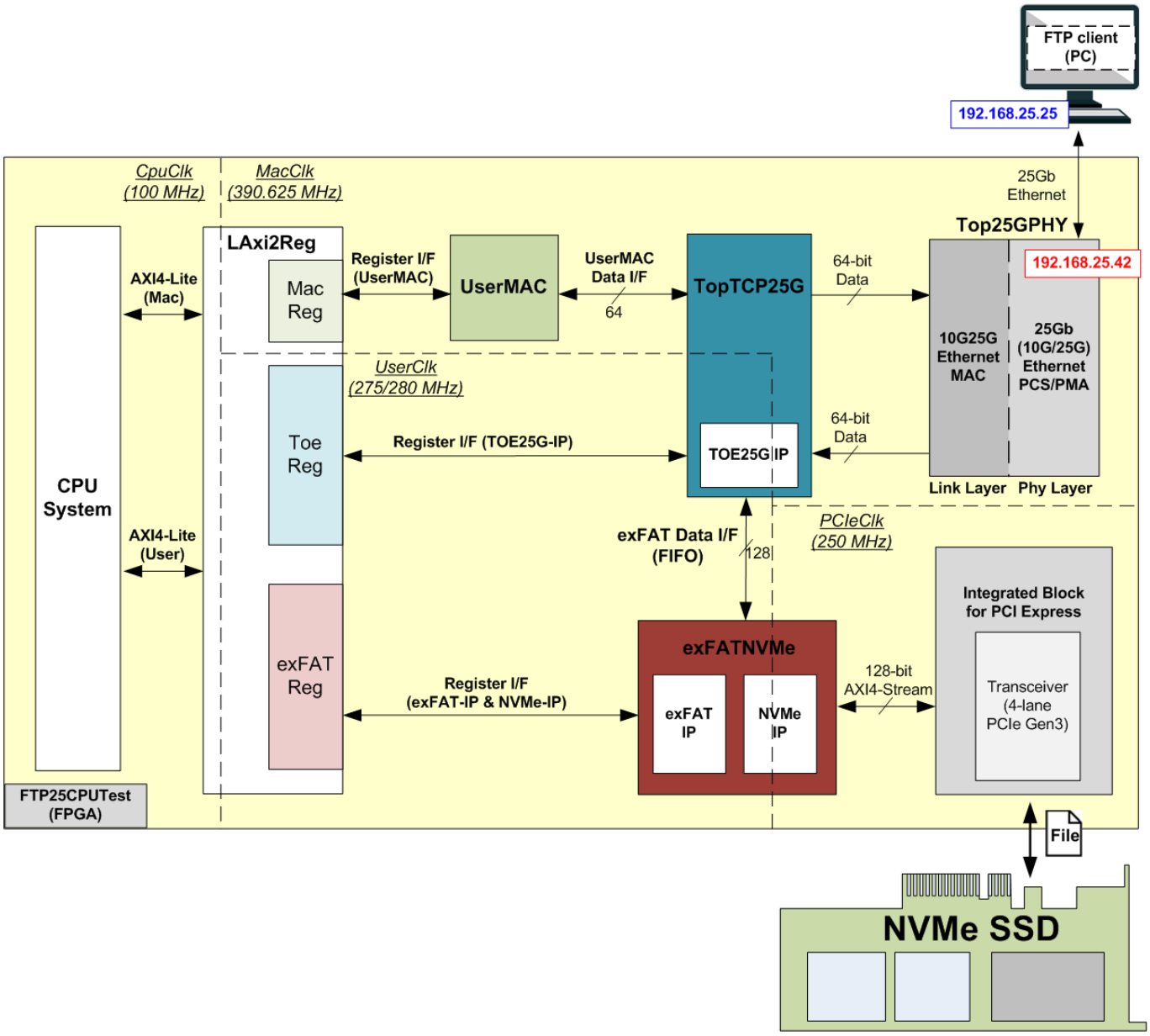

Figure 2-1 Demo block diagram

As described in the introduction, at least two connections are necessary for running FTP protocol, i.e., control connection for transferring FTP command and response and data connection for transferring data in some FTP commands such as LIST, STOR and RETR command.

The control connection is handled by CPU system. All data for control connection are created and monitored by CPU system through exFATReg, ToeReg and MacReg. When FTP client sends the new command, UserMAC forwards the command from 25G Ethernet MAC (10G25GEMAC) to CPU system for processing. After that, CPU returns FTP response to UserMAC which is forwarded to 10G25GEMAC. CPU system can access the hardware to monitor the system through two AXI4-Lite buses which are mapped to the different base addresses. One AXI4-Lite bus is for connecting with MacReg and another is for connecting with exFATReg and ToeReg which are mapped to different address.

The data connection is handled by TOE25G-IP which is implemented inside TopTCP25G. The data source of TOE25G-IP has two sources. First is file information, created by CPU system, when operating LIST command. Second is data in the requested file, read from NVMe SSD by exFATNVMe module, when operation RETR command. The received data in the data connection is directly connected to exFATNVMe module for operating STOR command.

By using TOE25G-IP with exFAT for NVMe system which consists of exFAT-IP and NVMe-IP, data transferring in FTP demo can achieve good performance on 25G Ethernet speed. Though exFAT for NVMe system performance is equal to NVMe PCIe Gen3 device speed (about 3300 MB/s), the performance in the demo is limited by 25 Gb Ethernet speed and personal computer specification.

More details of the hardware inside the demo are described in this topic.

2.1 25G (10G/25G) Ethernet PCS/PMA (25G BASE-SR)

This module implements PCS and PMA logics of 25G Ethernet. The physical interface is SFP28 for 25Gb BASE-SR standard. The user interface for connecting with 25G Ethernet MAC is 64-bit XGMII interface running at 390.625 MHz. This IP core can be created by using IP wizard in Vivado tools without the charge. More details of the core are described in the following link.

25G Ethernet PCS/PMA (BASE-SR)
https://www.xilinx.com/products/intellectual-property/ef-di-25gemac.html

2.2 Ethernet MAC (25G Ethernet)

In the reference design, DG 10G25GEMAC-IP is applied to connect between TOE25G-IP and PCS/PMA block for running 25Gb Ethernet application. The interface with TOE25G-IP is 64-bit AXI4-stream at 390.625 MHz while the interface with PCS/PMA is 64-bit XGMII at the same clock as AXI4-stream. More details about DG 10G25GEMAC-IP are described in the following link.
https://dgway.com/products/IP/10GEMAC-IP/dg_10g25gemacip_data_sheet_xilinx_en.pdf

2.3 TopTCP25G

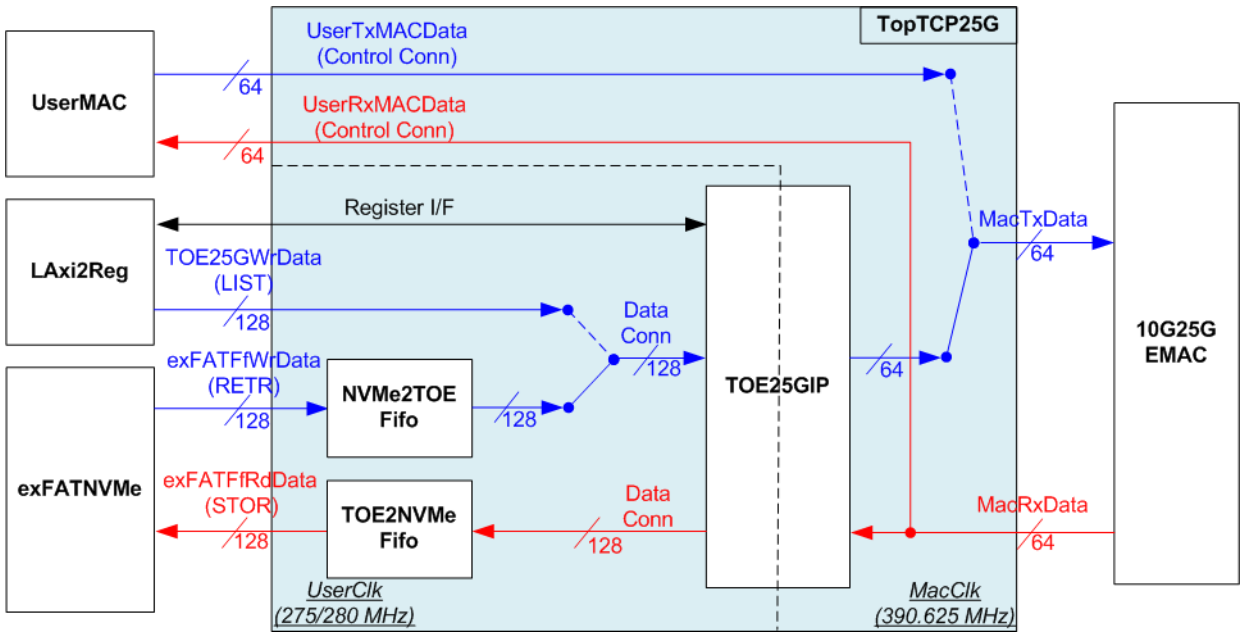

Figure 2-2 TopTCP25G block diagram

TopTCP25G is the module for transferring data with 25G Ethernet MAC (10G25GEMAC). To run FTP server application, two ports are implemented on the hardware. First is control connection and another is data connection. The control connection is responsible to decode FTP command and return FTP response. The data packet for control connection is small, so this port is implemented by using UserMAC controlled by CPU which is low-speed channel. The data connection is applied to transfer data of each file in RETR and STOR command and file list in LIST command. Since data size of data connection in RETR and STOR command is big, this port is implemented by using TOE25G-IP to achieve high-speed performance. The data source of TOE25G-IP is selected between LAxi2Reg and exFATNVMe. File list in LIST command is created by CPU through LAxi2Reg while data in the file when running RETR command is read from SSD by exFATNVMe.

After finishing the connection establishment, the port number of UserMAC is the port for control connection while the port number of TOE25G-IP is the port for data connection. Though all received packets from 10G25GEMAC-IP are forwarded to both UserMAC and TOE25G-IP, the packet filtering inside UserMAC and TOE25G-IP has never operated the same received packet by the different port number assignment.

As shown in Figure 2-2, there are two data-width user interfaces for TopTCP25G, i.e., 128-bit for TOE25G-IP and exFATNVMe which are run on UserClk domain and 64-bit for UserMAC which are run on MacClk, the same clock as EMAC. Moreover, there are 2 FIFOs included in TopTCP25G to act as the temporary buffers. Both buffers are applied to control the data flow between exFATNVMe and TOE25G-IP. NVMe2TOEFifo stores the read data returned from the SSD through exFATNVMe and then forwards to TOE25G-IP. TOE2NVMeFifo is designed to have the data direction reversed from NVMe2TOEFifo.

2.3.1 TOE25G-IP

TOE25G-IP implements TCP/IP stack and offload engine. User interface has two signal groups, i.e., control signals and data signals. Register interface is applied to set control registers and monitor status signals. Data signals are accessed by using FIFO interface. More details are described in datasheet.
http://www.dgway.com/products/IP/TOE25G-IP/dg_toe25gip_data_sheet_xilinx.pdf

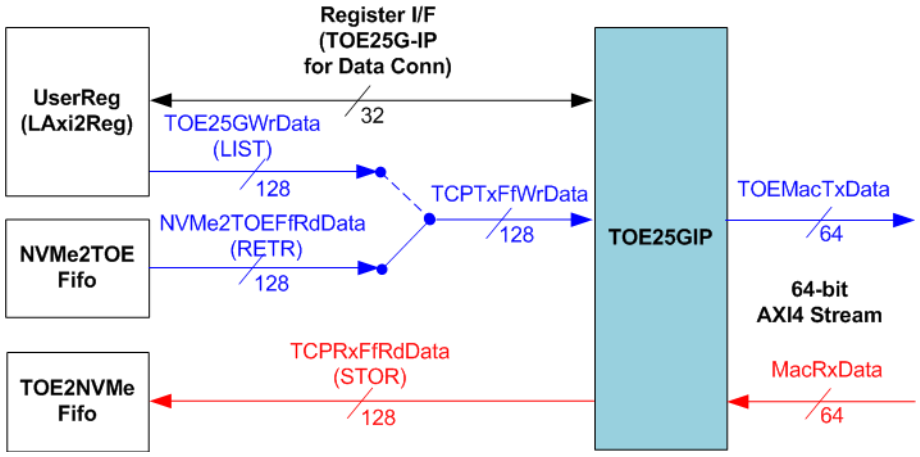

Figure 2-3 TOE25G-IP interface for data connection

In this demo, TOE25G-IP is applied for transferring data of data connection at high-speed rate. The network parameters of TOE25G-IP are set by CPU through LAxi2Reg to run as data connection of FTP command. Total transmit size of TOE25G-IP when running RETR command is also calculated and set by CPU which can be decoded from the file size in SSD. To operate STOR command, the received data of TOE25G-IP is stored to TOE2NVMeFifo when the receive buffer within TOE25G-IP is not empty and NVMe2TOEFifo is not full until completing the command.

2.3.2 Transmit data path (NVMe2TOEFifo)

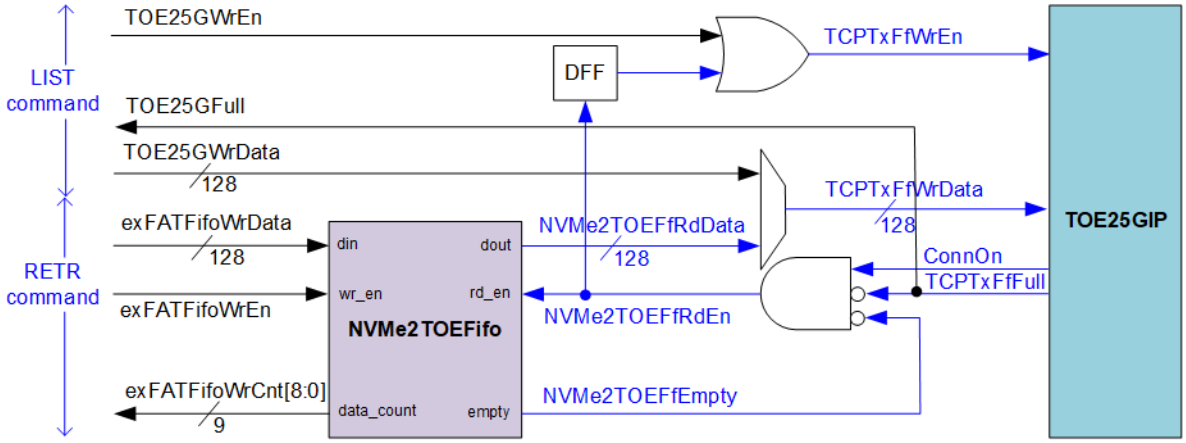

Figure 2-4 Transmit data path of TOE25G-IP

To transmit data to client through TOE25G-IP, there are two data sources, i.e., the list of file name stored in the SSD written by CPU firmware and the data of the file returned from the SSD through exFATNVMe, as shown in Figure 2-4.

To transfer the data of the file in RETR command, the step for transferring data from FIFO to TOE25G-IP is described as follows.

(1) The hardware must ensure that NVMe2TOEFifo has remaining data (NVMe2TOEFfEmpty='0'), Tx FIFO of TOE25G-IP is not full (TCPTxFfFull='0') and the data connection establishment is completed (ConnOn='1') before asserting read enable of NVMe2TOEFifo (NVMeTOEFfRdEn='1').

(2) In the next clock, the data transmits to TOE25G-IP by asserting TCPTxFfWrEn to '1' with the valid data from FIFO (NVMe2TOEFfRdData).

To transfer file name in LIST command, the design allows CPU to write data to TOE25G-IP directly by asserting TOE25GWrEn and assigning TOE25GWrData.

The FIFO is generated to control the data flow between exFATNVMe and TOE25G-IP. The size is 8 Kbytes (512x128-bit). exFATFifoWrCnt is monitored by exFATNVMe module to check the free space size in the FIFO. When the free space is enough and data of the SSD is ready, 512-byte data is sent to NVMe2TOEFifo.

2.3.3 Receive data path (TOE2NVMeFifo)

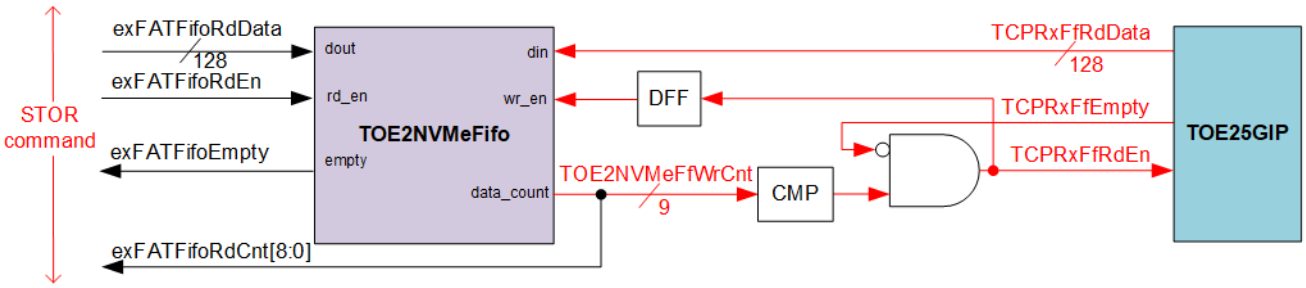

Figure 2-5 Receive data path of TOE25G-IP

The data read from client through TOE25G-IP is stored to TOE2NVMeFifo when operating STOR command. Data of the file received from FTP client is stored to the SSD through exFATNVMe. exFATFifoRdCnt and exFATFifoEmpty are monitored to check the available data in FIFO before forwarding to the SSD.

TOE2NVMeFifo is similar to NVMe2TOEFifo, the size is 8 Kbytes, but the data path is reversed.

2.4 UserMAC

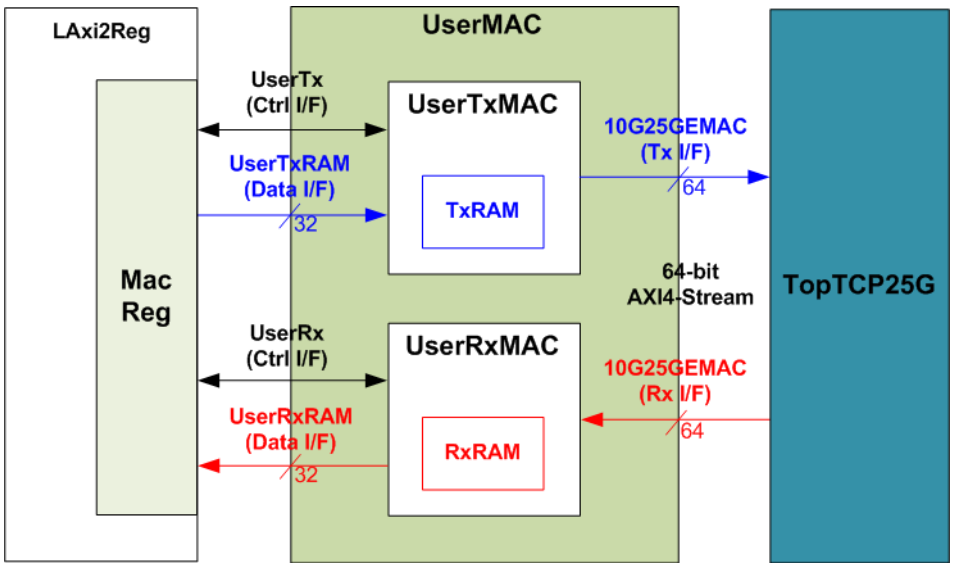

Figure 2-6 UserMAC block diagram

UserMAC is responsible to transfer TCP/IP packet for FTP control connection. The step of FTP control connection is designed by CPU firmware to access UserMAC registers through UserReg module. Data interface of UserMAC with UserReg is 32-bit RAM standard while data interface with 10G25GEMAC is 64-bit AXI4 Stream standard.

UserMAC consists of two modules, i.e., UserTxMAC and UserRxMAC. UserTxMAC includes TxRAM which stores the data written by CPU to 10G25GEMAC. UserRxMAC includes RxRAM which stores the received packet from 10G25GEMAC. The header of received packet has been verified by the packet filtering logic before stored to RxRAM. More details of UserTxMAC and UserRxMAC are described as follows.

2.4.1 UserTxMAC

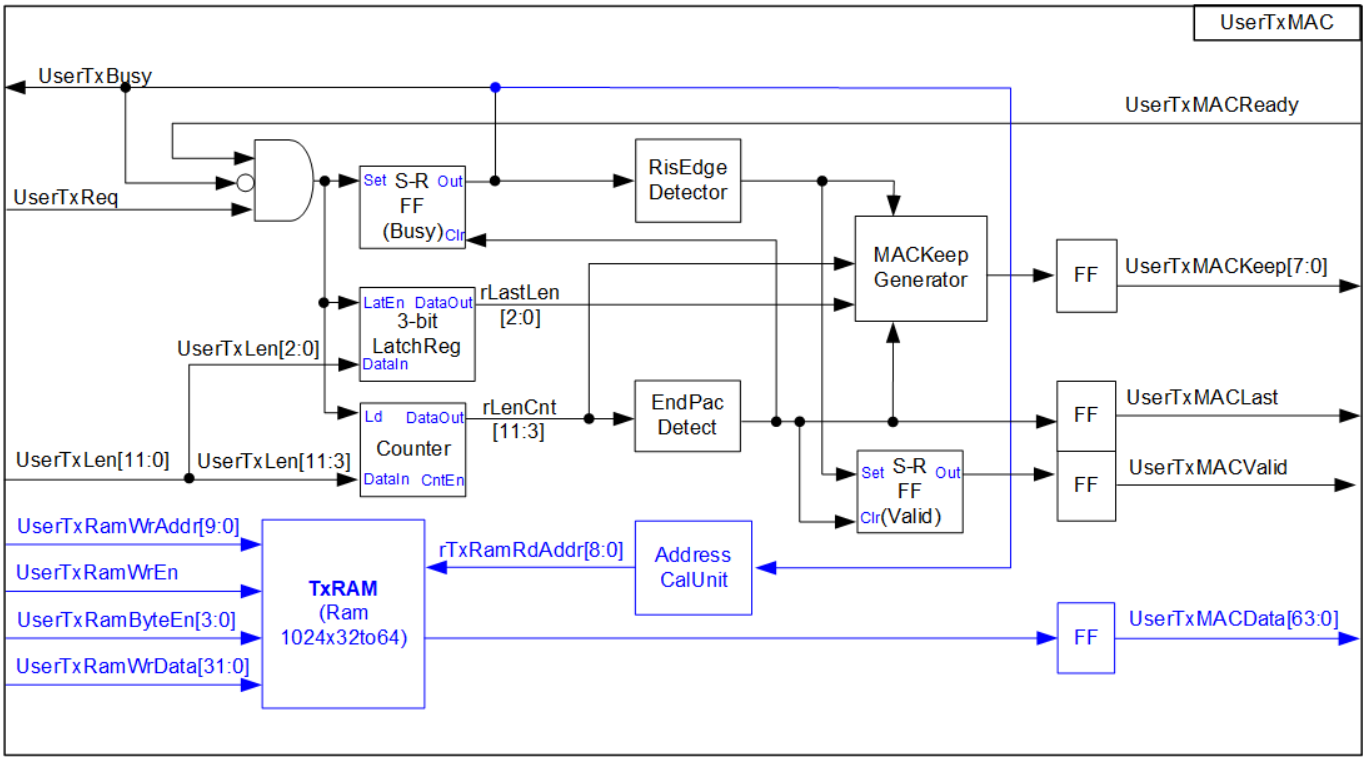

Figure 2-7 UserTxMAC logic diagram

The step to transfer the packet from TxRAM to EMAC is as follows.

(1) CPU confirms UserTxBusy='0' which means UserTxMAC is in Idle state.
(2) CPU writes transmit packet to TxRAM, starting from the 1st address (TxRamWrAddr=0).
(3) CPU sets UserTxLen=Packet size with asserting UserTxReq to '1' to begin data transmission.
(4) UserTxMAC waits until UserTxMACReady is asserted to '1' which means 10G25GEMAC is ready to receive the first data.
(5) UserTxBusy is asserted to '1' during transmitting data. At the same time, Counter loads total transfer size and starts counting to control packet size. AddressCalUnit resets the read address (rTxRamRdAddr) to 0 for reading the data starting from the 1st address. UserTxMACData is the read data output from TxRAM for sending to EMAC.
(6) UserTxMACValid is asserted to '1' to send the data along with UserTxMACData.
(7) The remaining data in the packet is continuously read from TxRAM and sent to 10G25GEMAC until end of the packet. At the same time, UserTxMACKeep is always set to all '1'.
(8) After the remaining packet size (rLenCnt) is equal to 1 which means the next data is the last data of the packet, UserTxMACKeep value reads rLastLen to generate the byte valid of the last data. It can be equal to 8 values: 0x01 (1-byte), 0x03 (2-byte), 0x07 (3-byte), 0x0F (4-byte), 0x1F (5-byte), 0x3F (6-byte), 0x7F (7-byte) or 0xFF (8-byte). UserTxMACLast is also asserted to '1' for sending the last data.
(9) After the last data of the packet is transmitted, UserTxMACValid and UserTxBusy are de-asserted to '0'.

2.4.2 UserRxMAC

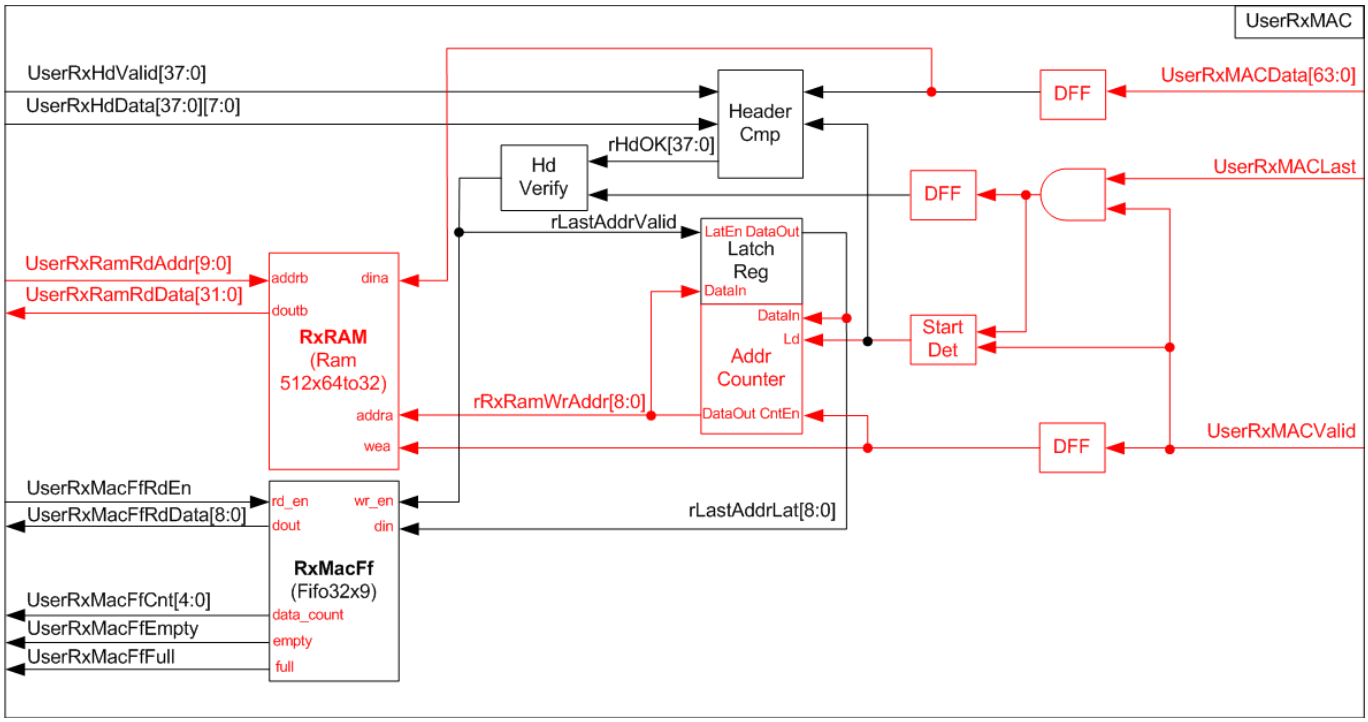

Figure 2-8 UserRxMAC logic diagram

The step when the new packet is received from EMAC is as follows.

(1) a) When the first valid data of the packet is detected, the enable signal to compare 38-byte header of each packet is asserted to '1' for 5 clock cycles (one clock cycle can compare 8-byte data).
b) At the same time, the address counter (AddrCounter) for generating write address of RxRAM (rRxRamWrAddr) loads the latest position of valid packet from rLastAddrLat. After that, the write address is increased to store the next data to the next address of RxRAM.
c) The received packet is stored to RxRAM by using UserRxMACValid to be write enable of RxRAM and UserRxMACData to be write data.

(2) Byte#0 – #37 of received data (UserRxMACData) are fed to header comparator module (HeaderCmp) to compare with the expected value (UserRxHdData), set by CPU firmware. UserRxHdValid is enable/disable flag to compare each byte of the header. Bit0, 1, 2, …, 37 of UserRxHdValid are the enable of byte0, 1, 2, …, 37 respectively.

(3) When the last data of the packet (UserRxMACLast) is detected, the result of header comparison flag (rHdOK) is read. If all headers are valid (rHdOK=all '1'), rLastAddrValid is asserted to '1'. At the same time, the latest address (rRxRamWrAddr) is loaded to rLastAddrLat for using in AddrCounter and RxMacFf. Otherwise, rLastAddrValid is not asserted to keep the same value of the latest valid address.

(4) CPU waits until the new packet is received by checking FIFO count (UserRxMacFfCnt) is not equal to 0. Next, CPU asserts read enable (UserRxMacFfRdEn) to read the end address which stores the last data of each packet (UserRxMacFfRdData). After that, CPU reads and decodes one received packet until the read address (UserRxRamRdAddr) is equal to the end address. Step (4) is repeated until FIFO count is equal to 0.
Note: UserRxRamRdAddr is the address for 32-bit data while rRxRamWrAddr is the address for 64-bit data. So, CPU firmware must convert the address of 64-bit data to the address of 32-bit data before starting reading data from RxRAM.

The 38-byte header of received packet for control connection of FTP application is shown as Figure 2-9.

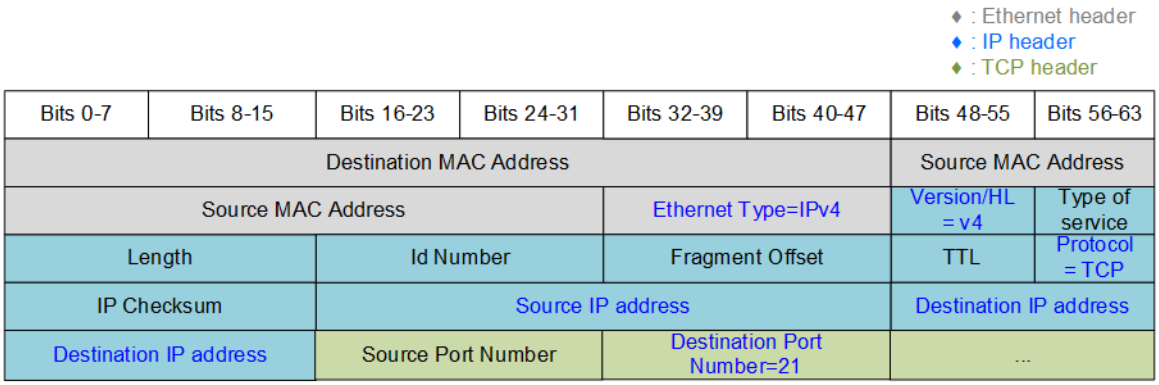

Figure 2-9 TCP/IP Packet header for FTP application

HeaderCmp is the packet filtering module which is controlled by CPU firmware. 38-byte header is verified with the data set from CPU. In the demo, CPU firmware assigns the enable flag and the expected data for comparing six data, described as follows (blue-color font in Figure 2-9).

1) Ethernet Type (2 bytes) = 0x0800 (IPv4)
2) IP version (1 byte) = 0x45 (Version 4)
3) Protocol (1 byte) = 0x06 (TCP Protocol)
4) Source IP Address (4 bytes) = IP address of FTP client
5) Destination IP Address (4 bytes) = IP address of FPGA
6) Destination Port Number (2 bytes) = 0x0015 (Port 21)

When the header of the packet is matched to the above parameters, the received packet is stored to RxRAM for CPU processing. Otherwise, the packet is ignored.

2.5 exFATNVMe

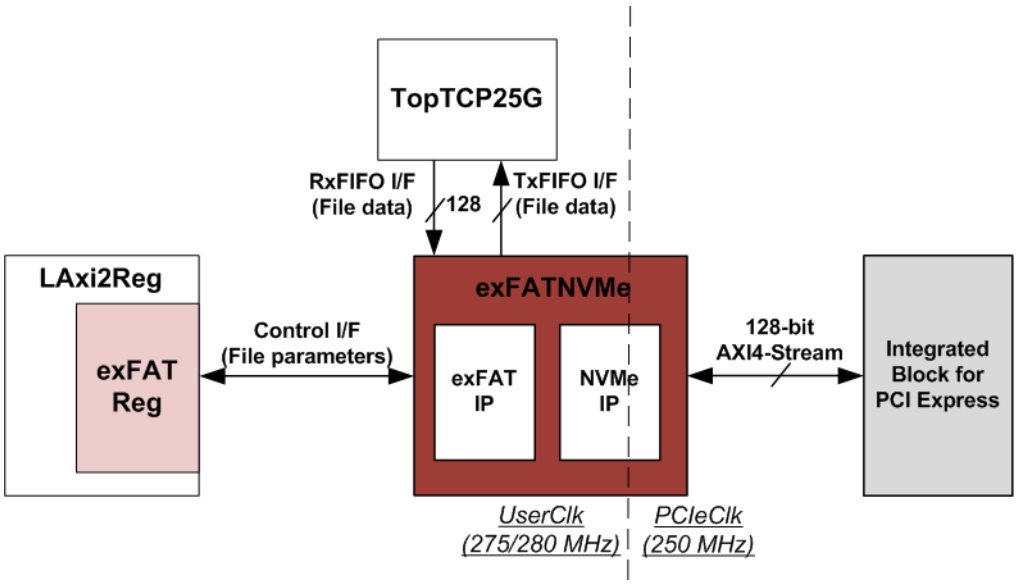

Figure 2-10 exFATNVMe block diagram

exFATNVMe is exFAT-IP core for NVMe SSD which consists of exFAT-IP and NVMe-IP. More details of exFAT-IP are described in the data sheet.
https://dgway.com/products/IP/NVMe-IP/dg_exfatip_nvme_data_sheet_en.pdf

In FTP 25G demo system, the control interface for setting file parameters is controlled by CPU through exFATReg inside LAxi2Reg module. The data interface (FIFO standard) is connected to TopTCP25G to transfer data with TOE25G-IP.

2.6 CPU and Peripherals

32-bit AXI4-Lite is applied to be the bus interface for the CPU accessing the peripherals such as Timer and UART. To control and monitor the test system, the control and status signals are connected to register for CPU access as a peripheral through 32-bit AXI4-Lite bus. CPU assigns the different base address and the address range to each peripheral for accessing one peripheral at a time.

In the reference design, the CPU system is built with two additional peripherals to access the hardwares in two clock domains. First clock is UserClk which is applied for controlling exFAT system and TOE25G-IP. Second clock is MacClk which is applied for controlling Ethernet MAC system. The base address and the range for accessing the hardware are defined in the CPU system. So, the hardware logic must be designed to support AXI4-Lite bus standard for supporting CPU writing and reading. LAxi2Reg module is designed to connect the CPU system as shown in Figure 2-11.

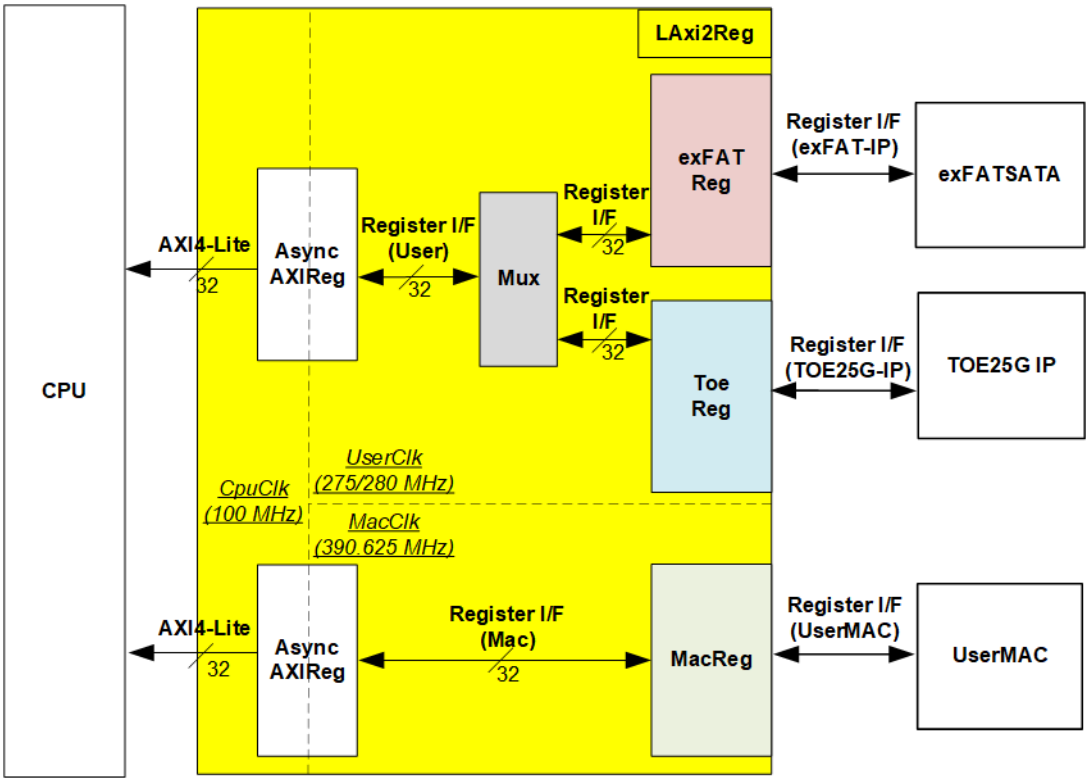

Figure 2-11 LAxi2Reg block diagram

LAxi2Reg consists of three modules: AsyncAXIReg, UserReg and exFATReg. The main responsibility of AsyncAXIReg is to convert 32-bit AXI4-Lite interface in CPU clock domain to register interface in Mac clock domain and User clock domain. AXI4-lite is the interface for CPU firmware communication with exFATReg, ToeReg and MacReg module. Two AsyncAXIReg modules are included in the system for supporting two clock domains. First clock domain is User clock which is used to operate TOE25G-IP and exFATNVMe, so the module requires a multiplexer for selecting the data between ToeReg and exFATReg to transfer through one AsyncAXIReg module. Another is Mac clock domain which is used to operate only UserMAC module. exFATReg, ToeReg and MacReg include the register file of the parameters and the status signals for exFATNVMe, TOE25G-IP and UserMAC respectively. More details of each module are described as follows.

2.6.1 AsyncAXIReg

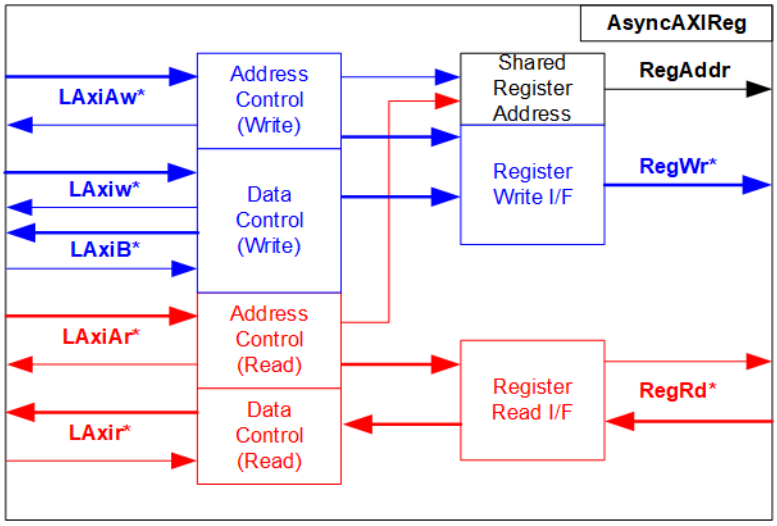

Figure 2-12 AsyncAXIReg interface

The signal on AXI4-Lite bus interface can be split into five groups: LAxiAw* (Write address channel), LAxiw* (Write data channel), LAxiB* (Write response channel), LAxiAr* (Read address channel) and LAxir* (Read data channel). More details to build custom logic for AXI4-Lite bus is described in following document.
https://forums.xilinx.com/xlnx/attachments/xlnx/NewUser/34911/1/designing_a_custom_axi_slave_rev1.pdf

According to AXI4-Lite standard, the write channel and the read channel are operated independently. Also, the control and data interface of each channel are run separately. So, the logic inside AsyncAxiReg to interface with AXI4-Lite bus is split into four groups: Write control logic, Write data logic, Read control logic and Read data logic as shown in the left side of Figure 2-12. Write control I/F and Write data I/F of AXI4-Lite bus are latched and transferred to be Write register interface with clock-crossing registers. In the same way, Read control I/F of AXI4-Lite bus are latched and transferred to be Read register interface with clock-crossing registers. After that, the returned data from Register Read I/F is transferred to AXI4-Lite bus by using clock-crossing registers. In register interface, RegAddr is shared signal for write and read access, so it loads the value from LAxiAw for write access or LAxiAr for read access.

The simple register interface is compatible with single-port RAM interface for write transaction. The read transaction of the register interface is slightly modified from RAM interface by adding RdReq and RdValid signals for controlling read latency time. The address of register interface is shared for write and read transaction, so user cannot write and read the register at the same time. The timing diagram of the register interface is shown in Figure 2-13.

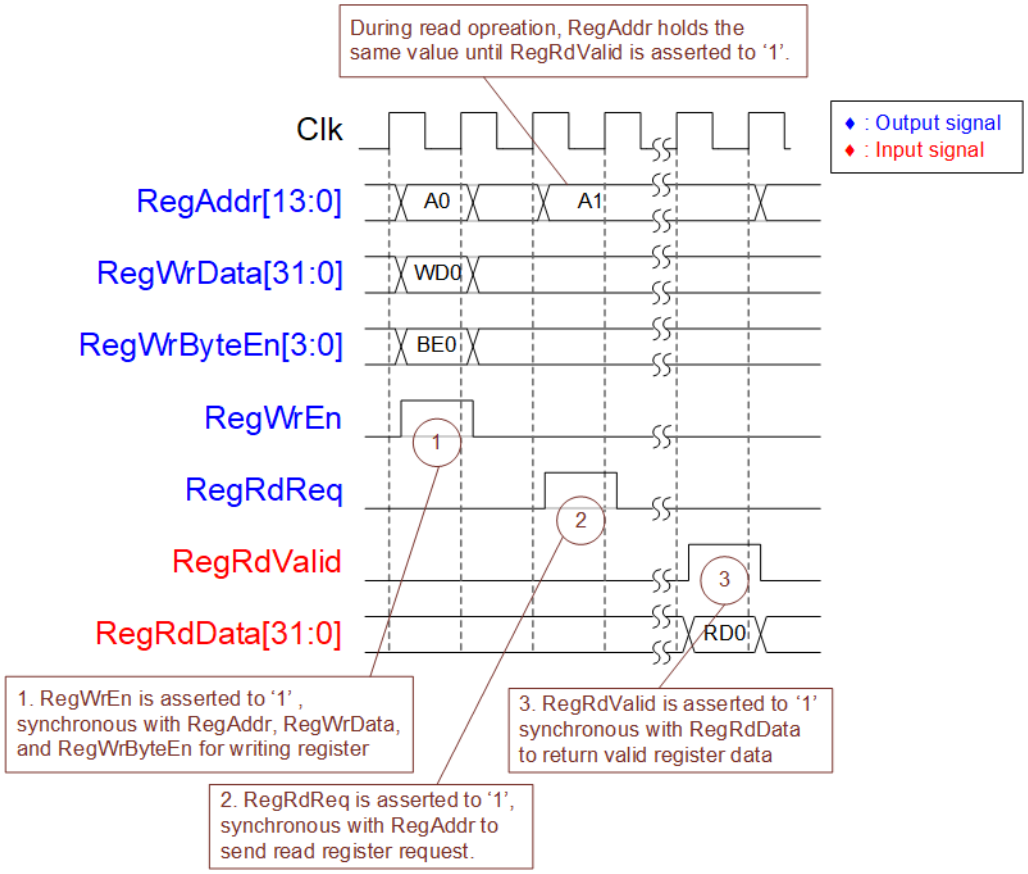

Figure 2-13 Register interface timing diagram

1) To write register, the timing diagram is similar to single-port RAM interface. RegWrEn is asserted to '1' with the valid signal of RegAddr (Register address in 32-bit unit), RegWrData (write data of the register) and RegWrByteEn (the write byte enable). Byte enable has four bits to be the byte data valid. Bit[0], [1], [2] and [3] are equal to '1' when RegWrData[7:0], [15:8], [23:16] and [31:24] are valid respectively.
2) To read register, AsyncAxiReg asserts RegRdReq to '1' with the valid value of RegAddr. 32-bit data must be returned after receiving the read request. The slave must monitor RegRdReq signal to start the read transaction. During read operation, the address value (RegAddr) does not change the value until RegRdValid is asserted to '1'. So, the address can be used for selecting the returned data by using multiple layers of multiplexer.
3) The read data is returned on RegRdData bus by the slave with asserting RegRdValid to '1'. After that, AsyncAxiReg forwards the read value to LAxir* interface.

2.6.2 exFATReg

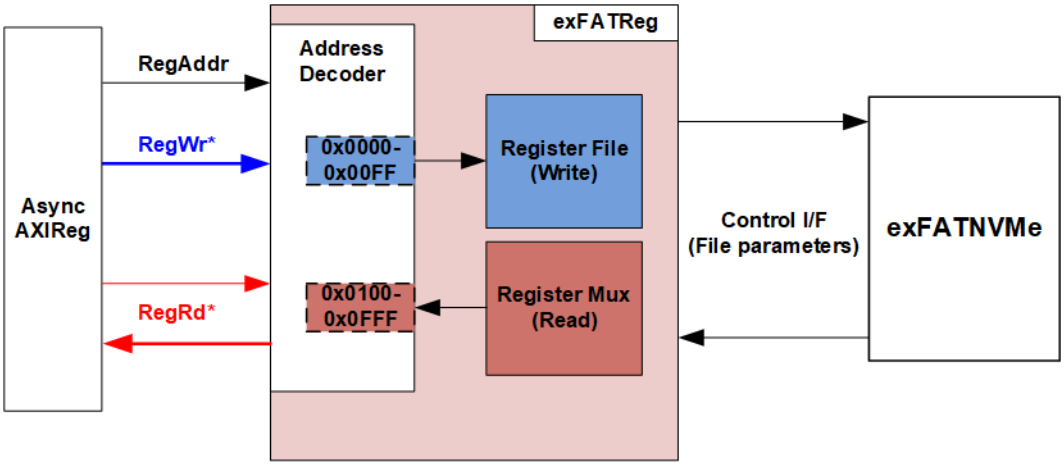

Figure 2-14 exFATReg block diagram

The address range to map to exFATReg is split into two areas, as shown in Figure 2-14.

1) 0x0000 – 0x00FF: mapped to set file parameters of exFATNVMe module. This area is write-access only.
2) 0x0100 – 0x0FFF: mapped to read the status of exFATNVMe module. This area is read-access only.

Address decoder decodes the upper bit of RegAddr for selecting the active hardware. The latency of read data is equal to two clock cycles, so RegRdValid is created by RegRdReq with asserting two D Flip-flops. More details of the address mapping within exFATReg module is shown in Table 2-1.

Table 2-1 Memory Map of exFATReg

Address Wr/Rd	Register Name Label in “ftp25cputest.c”	Description
BA0+0x0000 – BA0+0x00FF: Control signals of exFAT-IP (Write access only)		
BA0+0x0000	User File Name Reg (USRFNAME_REG)	[26:0] - Input to be UserFName of exFAT-IP for NVMe
BA0+0x0004	User File Length Reg (USRFLEN_REG)	[26:0] - Input to be UserFLen of exFAT-IP for NVMe
BA0+0x0008	User File Size Reg (USRFSIZE_REG)	[2:0] - Input to be FSize of exFAT-IP for NVMe. Set File size to exFAT-IP when running Format command.
BA0+0x000C	Date and Time Reg (DATETIME_REG)	[4:0] - Input to be FTimeS of exFAT-IP for NVMe [10:5] - Input to be FTimeM of exFAT-IP for NVMe [15:11] - Input to be FTimeH of exFAT-IP for NVMe [20:16] - Input to be FDateD of exFAT-IP for NVMe [24:21] - Input to be FDateM of exFAT-IP for NVMe [31:25] - Input to be FDateY of exFAT-IP for NVMe
BA0+0x0010	exFAT Command Reg (EXFATCMD_REG)	[1:0] - Input to be UserCmd of exFAT-IP for NVMe When this register is written, exFATIPReg is asserted to ’1’ to start new command operation.
BA0+0x0100 – BA0+0x08FF: Status signals of exFAT-IP and NVMe-IP (Read access only)		
BA0+0x0100	exFAT status Reg (FATSTS_REG)	[0] - Mapped to UserBusy of exFAT-IP for NVMe [1] - Mapped to UserError of exFAT-IP for NVMe
BA0+0x0104	Total file capacity Reg (TOTALFCAP_REG)	[26:0] - Mapped to TotalFCap[26:0] of exFAT-IP for NVMe
BA0+0x0108	User error type Reg (FATERRTYPE_REG)	[31:0] - Mapped to UserErrorType[31:0] of exFAT-IP for NVMe
BA0+0x010C	exFAT IP test pin (Low) Reg (FATTESTPINL_REG)	[31:0] - Mapped to TestPin[31:0] of exFAT-IP for NVMe
BA0+0x0110	exFAT IP test pin (High) Reg (FATTESTPINH_REG)	[31:0] - Mapped to TestPin[63:32] of exFAT-IP for NVMe
BA0+0x0114	Directory capacity Reg (DIRCAP_REG)	[19:0] - Mapped to DirCap[19:0] of exFAT-IP for NVMe
BA0+0x0118	File size in the disk Reg (DFSIZE_REG)	[2:0] - Mapped to DiskFsize of exFAT-IP for NVMe. This register shows the current file size used in the device, read by exFAT-IP for NVMe.
BA0+0x011C	Total file in the disk Reg (DFNUM_REG)	[26:0] - Mapped to DiskFnum of exFAT-IP for NVMe
BA0+0x0120	Disk capacity (Low) Reg (DCAPL_REG)	[31:0]: Mapped to LBASize(bit[31:0]) of NVMe-IP to check total capacity of the NVMe device
BA0+0x0124	Disk capacity (High) reg (DCAPH_REG)	[15:0]: Mapped to LBASize(bit[47:32]) of NVMe-IP to check total capacity of the NVMe device
BA0+0x0128	Completion status Reg (COMPSTS_REG)	[15:0]: Status from Admin completion (AdmCompStatus[15:0] of NVMe-IP) [31:16]: Status from I/O completion (IOCompStatus[15:0] of NVMe-IP)
BA0+0x012C	NVMe CAP Reg (NVMCAP_REG)	[31:0]: Mapped to NVMeCAPReg [31:0] of NVMe-IP
BA0+0x0130	NVMe test pin Reg (NVMTESTPIN_REG)	[31:0]: Mapped to TestPin[31:0] of NVMe-IP
BA0+0x0200	Current transfer size (Low) Reg (FATCURTRNSIZEL_REG)	[31:0]: Bit[31:0] of the current transfer byte size in exFATNVMe module
BA0+0x0204	Current transfer size (High) Reg (FATCURTRNSIZEH_REG)	[24:0]: Bit[56:32] of the current transfer byte size in exFATNVMe module
BA0+0x0800	exFAT-IP Version Reg (EXFATNVME_VER_REG)	[31:0]: exFAT-IP version number, mapped to IPVersion [31:0] of exFAT-IP
BA0+0x0804	NVMe IP Version Reg (NVME_VER_REG)	[31:0]: NVMe-IP version number, mapped to IPVersion [31:0] of NVMe-IP

2.6.3 ToeReg

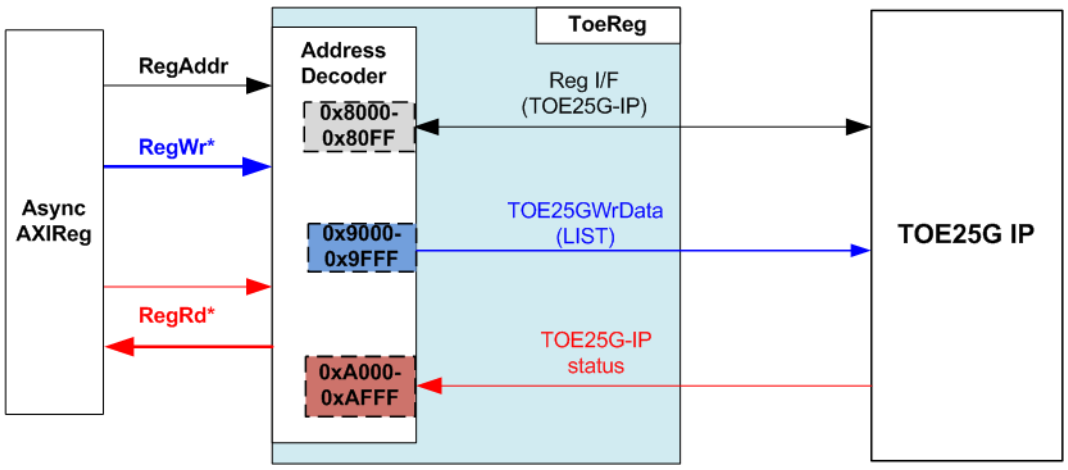

Figure 2-15 ToeReg block diagram

ToeReg includes the registers of TOE25G-IP. The address range is split into three areas.

1) *0x8000 – 0x80FF:* mapped to TOE25G-IP register for setting and monitoring network parameters and control signals.
2) *0x9000 – 0x9FFF:* mapped to write TCP data which is the list of file name, sent to TOE25G-IP for LIST command.
3) *0xA000 – 0xAFFF:* mapped to read the status signal, output from TOE25G-IP.

Address decoder decodes the upper bit of RegAddr for selecting the active hardware. The latency of read data is one clock cycle, so RegRdValid is created by RegRdReq with asserting one D Flip-flop. More details of the address mapping within ToeReg module is shown in Table 2-2.

Table 2-2 Memory Map of ToeReg

Address	Register Name	Description
Wr/Rd	Label in “ftp25cputest.c”	
BA0+0x8000 - BA0+0x80FF: TOE25G-IP Register Area More details of each register are described in TOE25G-IP datasheet		
BA0+0x8000	TOE_RST_REG	Mapped to RST register within TOE25G-IP
BA0+0x8004	TOE_CMD_REG	Mapped to CMD register within TOE25G-IP
BA0+0x8008	TOE_SML_REG	Mapped to SML register within TOE25G-IP
BA0+0x800C	TOE_SMH_REG	Mapped to SMH register within TOE25G-IP
BA0+0x8010	TOE_DIP_REG	Mapped to DIP register within TOE25G-IP
BA0+0x8014	TOE_SIP_REG	Mapped to SIP register within TOE25G-IP
BA0+0x8018	TOE_DPN_REG	Mapped to DPN register within TOE25G-IP
BA0+0x801C	TOE_SPN_REG	Mapped to SPN register within TOE25G-IP
BA0+0x8020	TOE_TDL_REG	Mapped to TDL register within TOE25G-IP
BA0+0x8024	TOE_TMO_REG	Mapped to TMO register within TOE25G-IP
BA0+0x8028	TOE_PKL_REG	Mapped to PKL register within TOE25G-IP
BA0+0x802C	TOE_PSH_REG	Mapped to PSH register within TOE25G-IP
BA0+0x8030	TOE_WIN_REG	Mapped to WIN register within TOE25G-IP
BA0+0x8034	TOE_ETL_REG	Mapped to ETL register within TOE25G-IP
BA0+0x8038	TOE_SRV_REG	Mapped to SRV register within TOE25G-IP
BA0+0x803C	TOE_VER_REG	Mapped to VER register within TOE25G-IP
BA0+0x9000 - BA0+0x9FFF: Tx data interface for TOE25G-IP		
BA0+0x9000 Wr	Write data to TOE25G-IP (TOE_TXDATA_REG0)	[31:0]: Bit[31:0] of Transmitted data written by CPU for sending through TOE25G-IP
BA0+0x9004 Wr	Write data to TOE25G-IP (TOE_TXDATA_REG1)	[31:0]: Bit[63:32] of Transmitted data written by CPU for sending through TOE25G-IP
BA0+0x9008 Wr	Write data to TOE25G-IP (TOE_TXDATA_REG2)	[31:0]: Bit[95:64] of Transmitted data written by CPU for sending through TOE25G-IP
BA0+0x900C Wr	Write data to TOE25G-IP (TOE_TXDATA_REG3)	[31:0]: Bit[128:96] of Transmitted data written by CPU for sending through TOE25G-IP. When this register is written, it generates a pulse to write one 128-bit data to TOE25G-IP
BA0+0xA000 - BA0+0xAFFF: TOE25G-IP Status Area		
BA0+0xA000 Rd	TOE25G Status Register (TOE_STS_REG)	[0] - Reserved [1] - Mapped to ConnOn signal of TOE25G-IP [2] - Mapped to TCPTxFfFull signal of TOE25G-IP
BA0+0xA004 Wr/Rd	Interrupt Status of Ethernet interface (ETHER_INT_STS)	Wr: bit[0]: Set ‘1’ to clear TimerInt latch value Rd: bit[0]- ‘1’: Detect TimerInt from TOE25G-IP, ‘0’: No interrupt

2.6.4 MacReg

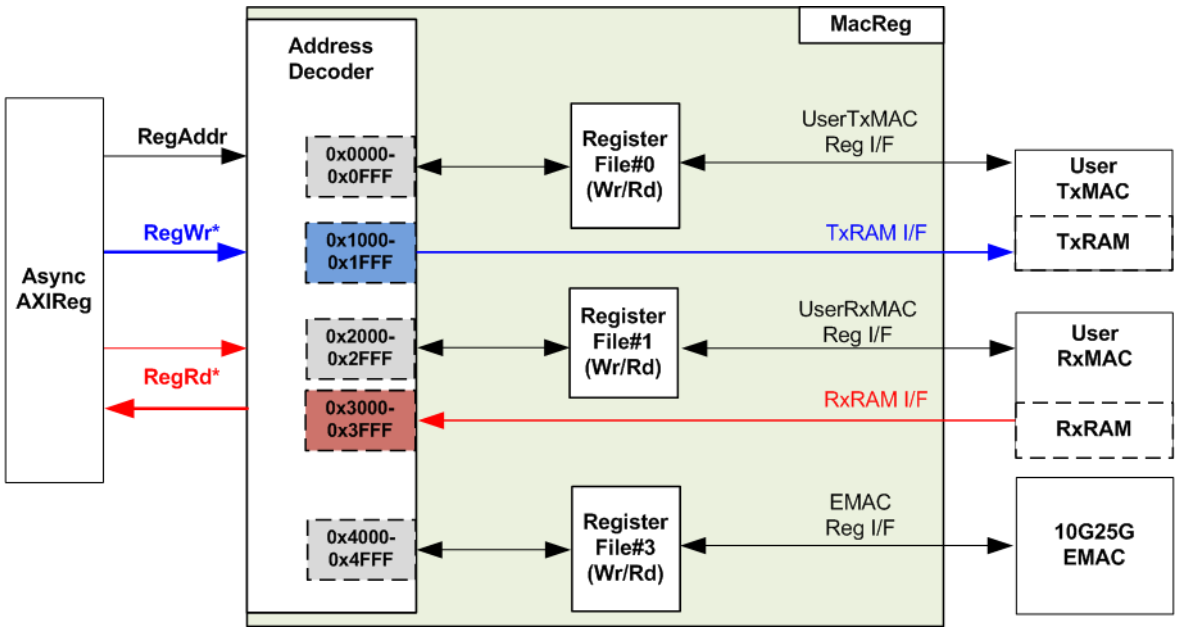

Figure 2-16 MacReg Block diagram

MacReg includes the registers of the module for processing Ethernet packet such as UserTxMAC and UserRxMAC. The address range is split into five areas.

1) *0x0000 – 0x0FFF:* mapped to the control/status signals of UserTxMAC for transmitting packet.
2) *0x1000 – 0x1FFF:* mapped to write TxRAM which stores transmitted packet of FTP control connection. This area is write-access only.
3) *0x2000 – 0x2FFF:* mapped to read RxRAM which stores received packet of FTP control connection. This area is read-access only.
4) *0x3000 – 0x3FFF:* mapped to the control/status signals of UserRxMAC for receiving packet.
5) *0x4000 – 0x4FFF:* mapped to the control/status signals of DG 10G25GEMAC-IP

Address decoder decodes the upper bit of RegAddr for selecting the active hardware. To read register, three-step multiplexers are designed to select the read data, returned to CPU. The latency of read data is equal to three clock cycles, so RegRdValid is created by RegRdReq with asserting three D Flip-flops. More details of the address mapping within MacReg module is shown in Table 2-3.

Table 2-3 Memory Map of MacReg

Address	Register Name	Description
Wr/Rd	Label in “ftp25cputest.c”	
BA1+0x0000 – BA1+0x1FFF: UserTxMAC Area		
BA1+0x0000	Total Transmit Length	Wr [11:0] - Total transmit size in byte unit. Valid from 1-0xFFF. UserTxMAC starts transmitting packet after this register is written. Rd [0] – Busy flag of UserTxMAC. ‘0’-Idle, ‘1’-Busy.
Wr/Rd	(TXEMAC_LEN_REG)	
BA1+0x0000 – BA1+0x1FFF	TxRAM in UserTxMAC	Transmitted data written by CPU for sending through UserTxMAC
Wr	(TXRAM_BASE_ADDR)	
BA1+0x2000 – BA1+0x3FFF: UserRxMAC Area		
BA1+0x2000 – BA1+0x2024	UserRxMAC Header Data	38-byte to set the packet filter inside UserRxMAC for comparing byte#0 – byte#37 of received packet when header byte enable is asserted to ‘1’. If header byte enable is de-asserted to ‘0’, the filter bypasses that byte. *0x5000[7:0] – byte#0, [15:8] – byte#1, [23:16] – byte#2, [31:24] – byte#3* *0x5004[7:0] – byte#4, [15:8] – byte#5, [23:16] – byte#6, [31:24] – byte#7* *…* *0x5020[7:0] – byte#32, [15:8] – byte#33, [23:16] – byte#34, [31:24] – byte#35* *0x5024[7:0] – byte#36, [15:8] – byte#37*
Wr	(RXEMAC_HDVAL)	
BA1+0x2028 – BA1+0x202C	UserRxMAC Header Byte Enable	38-bit to compare 38-byte header of received packet in UserRxMAC *0x50028[0]* – Compare enable of *byte#0, [1] – byte#1, …, [31] – byte#31* *0x5002C[0] – byte#32, [1] – byte#33, .., [5] – byte#37*
Wr	(RXEMAC_HDEN)	
BA1+0x2040	RxEMAC Last Address	Rd [8:0] – Read last address from RxMacFf When reading register, it generates a pulse of read enable for reading one data from RxMacFf. So, please check that there is data in RxMacFf before reading this register by checking RXEMAC_FFCNT is not equal to 0.
Rd	(RXEMAC_LASTADDR)	
BA1+0x2044	FIFO Counter of RxMacFf	Rd [4:0] – FIFO counter to show total number of data in RxMacFf.
Rd	(RXEMAC_FFCNT)	
BA1+0x3000 – BA1+0x3FFF	RxRAM in UserRxMAC	Received data stored in RxRAM within UserRxMAC
Rd	(RXRAM_BASE_ADDR)	
BA1+0x4000 – BA1+0x4FFF: EMAC Status Area		
BA1+0x4000	Interrupt Status of Ethernet interface	Wr: [0]-Set ‘1’ to clear interrupt when RxMacFf is full Rd: [0]-‘1’: Detect Full flag of RxMacFf asserted, ‘0’: Not full
Wr/Rd	(ETHER_INT_STS)	
BA1+0x4004	EMAC Linkup status	[0]: Ethernet linkup status from Ethernet MAC (‘0’: Not linkup, ‘1’: Linkup)
Rd	(EMAC_LINKUP_STS)	
BA1+0x4008	EMAC-IP Version Reg	Rd[31:0] – Mapped to IPVersion output from DG 10G25GEMAC-IP when the system integrates DG 10G25GEMAC-IP.
Rd	(EMAC_VER_REG)	

3 CPU Firmware

CPU firmware implements two functions for running ftp server demo. First, CPU handles TCP/IP packet of FTP control connection through MacReg and UserMAC module which transmits and receives TCP/IP packet of control connection. Second, CPU controls and monitors the registers of exFATNVMe and TOE25G-IP for handling TCP/IP packet of FTP data connection.

To run FTP server demo, CPU firmware operation is shown in Figure 3-1.

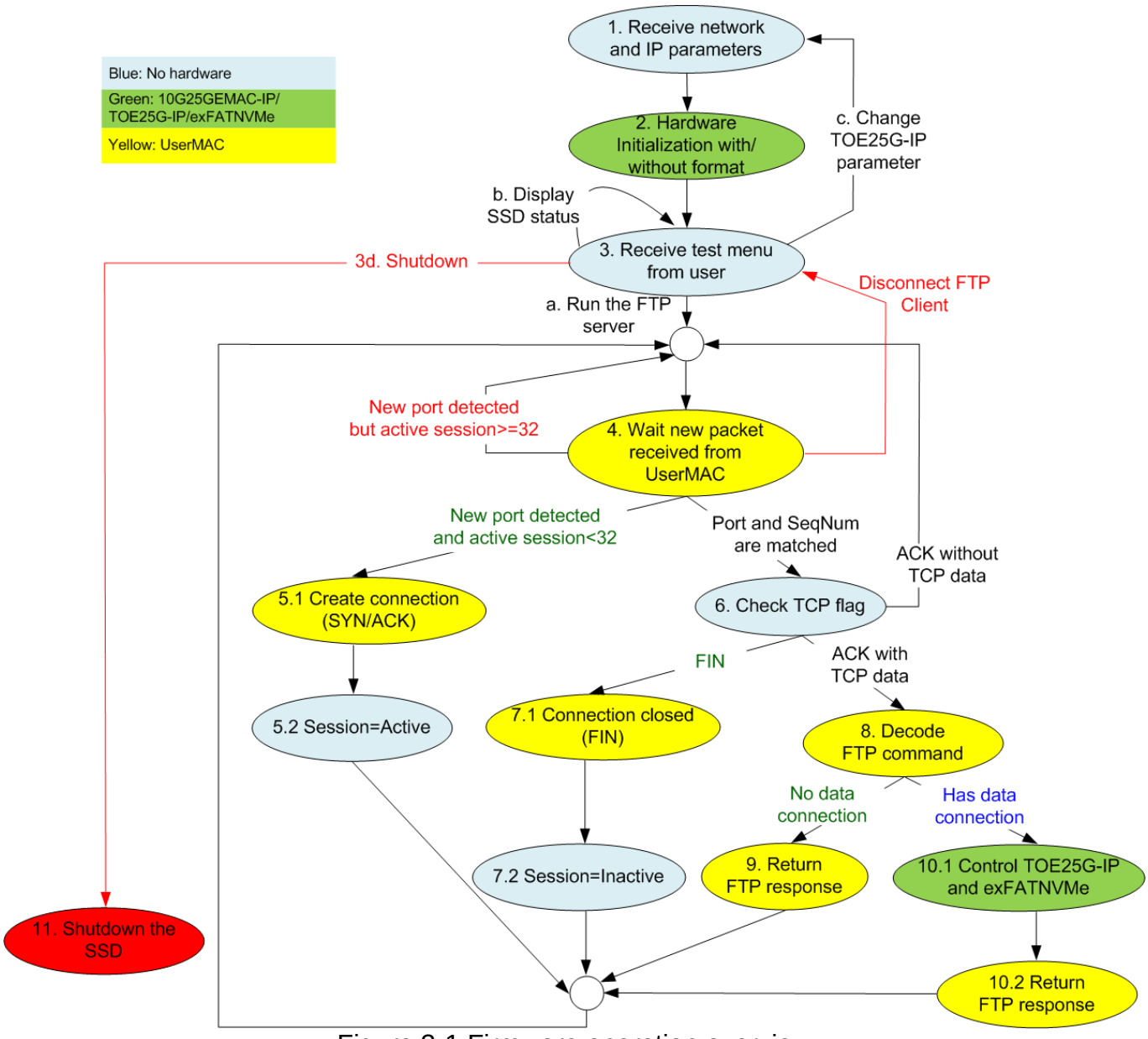

Figure 3-1 Firmware operation overview

1 Receive and validate the network and IP parameters to initialize DG 10G25GEMAC-IP, TOE25G-IP and the SSD. Then, set the parameters to TOE25G-IP, UserRxMAC and exFAT-IP.
2 CPU waits until the hardware finishes the initialization process by monitoring linkup status of 10G25GEMAC-IP and busy flag of TOE25G-IP (TOE_CMD_REG) and exFAT-IP (FATSTS_REG).
3 Enter to main menu for selecting the test mode. Four options are displayed as follows.
   a) Run the FTP server. This menu starts receiving FTP packet from client. This menu is run as forever loop. User must enter special keys to return to main menu. Next, continue to step 4.
   b) Change file created time and display the SSD information such as current file size (DFSIZE_REG), maximum file in the disk (TOTALFCAP_REG) and total file in the disk (DFNUM_REG). After displaying SSD information is done, CPU returns to main menu.
   c) Change network parameters such as IP address, MAC address and port number. After changing network parameters is done, CPU returns to main menu.
   d) Shutdown the server by sending the shutdown command to the SSD. Busy flag (FAT_STS_BUSY[0]) is de-asserted to '0' after the shutdown process is completed. The SSD and CPU change to inactive status and the system cannot receive any command from user. Next, continue to step 11.
4 CPU waits until the new packet is received from UserRxMAC. When the new packet is detected, CPU decodes TCP flag to select the next step.
   a) When the flag is SYN and total session now is less than 32, CPU goes to step 5.
   b) If total session is equal to 32 which means that maximum session is reached, CPU stays in this step and waits for some sessions to be closed.
   c) Otherwise, CPU goes to step 6.
5 Create the new connection by running following step.
   5.1 Return SYN/ACK packet and then wait until receiving the valid ACK without data packet.
   5.2 Change the session status from Inactive to Active.
6 CPU finds the active session which the port number is matched and verifies the sequence number is correct value.
   a) When the flag is FIN, CPU goes to step 7.
   b) When the flag is ACK and TCP data length is not equal to 0, CPU goes to step 8.
   c) Otherwise, CPU has no operation and goes back to step 4.
7 Terminate the connection by running the following step.
   7.1 Send FIN/ACK packet to close the connection and then wait until the valid ACK without data is received.
   7.2 Change the session status from Active to Inactive.
8 CPU decodes FTP command. There are two command types, i.e., command without data connection and command with data connection.
   a) CPU goes to step 9 for processing command without data connection.
   b) Otherwise, CPU goes to step 10.
9 CPU prepares FTP response of each FTP command to TxRAM and then sets the parameters to start transmitting FTP response to UserTxMAC.
   More details when running FTP command without data connection are described in topic 3.1.
10 Data connection and control connection are run as following step.
   10.1 TOE25G-IP and exFATNVMe registers are set and monitored by CPU to start data transferring on FTP data connection. Then, wait until finishing data transmission.
   10.2 Return FTP response, similar to step 9.
   More details when running FTP command with data connection are described in topic 3.2.
11 Send Shutdown command to SSD and hold on until the system is reset.

3.1 FTP command without data connection

The step when the new FTP command is received and CPU returns response on control connection is as follows.

1) Wait until new received packet is stored in RxRAM by monitoring data counter of RxMacFIFO (RXEMAC_FFCNT) is not equal to 0.
2) Read data from RxMacFIFO by reading RXEMAC_LASTADDR. After that, CPU copies data from RxRAM to the temporary buffer on firmware, beginning from the latest position to RXEMAC_LASTADDR value.
   Note: The temporary buffer on firmware is defined by BUFFSIZE parameter which is equal to 128. So, the demo without modification can support up to 128-byte packet size which is enough for processing FTP control connection.
3) Update the latest position in the firmware to be the new value (RXEMAC_LASTADDR).
4) Find the active port which has the client port equal to the source port in the received packet.
   a) Continue to the next step when the port is matched, the sequence number in the received packet is equal to the expected value and received TCP flag is ACK without SYN and FIN flag.
   b) Otherwise, CPU creates or destroys the session when receiving SYN flag or FIN flag following three-way handshake respectively.
5) Continue the next step when TCP data length in the received packet is not equal to 0. TCP data length is calculated by (IP length – IP header length – TCP header length).
6) Create FTP response packet to the transmit temporary buffer. The value of FPGA response is created following FTP command. Since there are a lot of standard FTP commands, the demo implements some mandatory commands, related to FTP client application, FileZilla. Lists of implemented command without data connection are described in Table 3-1.
7) Call function to prepare the header and calculate checksum for transmitting FTP response.
8) Copy the packet from transmit temporary buffer to TxRAM and set total length to TXEMAC_LEN_REG for starting packet transmission.
9) Wait until the packet transmission is finished by detecting when busy flag of UserTxMAC (TXEMAC_LEN_REG) is equal to 0.
10) Go back to step 1) for processing the next FTP command.

Table 3-1 FTP Response for FTP command without data connection

FTP command	Description	Implemented FTP Response
USER	Authentication username	331 User is correct. Password is required
PASS	Authentication password	230 Logged in
		530 Login is incorrect
PWD	Print working directory	257 "PATHNAME" is the current directory
TYPE	Set the transfer mode	200 Type set to I
PASV	Enter passive mode	227 Enter Passive mode (h1,h2,h3,h4,p1,p2) h1-h4: IP address, p1-p2: Port number
CWD	Change working directory	250 Requested file action Okay
DELE	Delete file	202 No implemented
QUIT	Log out session	221 Bye.

When the command is not in the above list such as AUTH, PORT, SYST and FEAT, FTP response 500 (syntax error) is returned. PASV command is the command sent by client before sending FTP command with data connection, i.e., LIST, RETR, and STOR. More details of FTP command with data connection are described in the next topic.

Note: *DELE command is not implemented in the reference design, so user cannot delete any files from the server's storage.*

3.2 FTP command with data connection

Table 3-2 FTP Response for FTP command with data connection

FTP command	Description	Implemented FTP Response
LIST	Return File list in the current directory	125 Open data connection 226 Transferring complete
STOR	Accept the data and store the data as a file at the server site	
RETR	Retrieve a copy of the file	

In the demo, three FTP commands using data connection are implemented as shown in Table 3-2. The details of each FTP command are described as follows.

3.2.1 LIST

This command is applied to return file list in the current directory. List of files is created by CPU firmware and the packet must be returned in FTP data connection which is controlled by TOE25G-IP. So, CPU prepares file list and sends to TOE25G-IP. The step to run LIST command is described as follows.

1) Read the number of files in the device from DFNUM_REG.
2) Wait until data connection establishment completes by monitoring ConnOn of TOE25G-IP (TOEIP_STS_REG[1]='1').
3) Call function to prepare FTP response 125 to TxRAM and start packet transmission.
4) Set TOE25G-IP parameters for sending file information, i.e., transfer packet size (TOE_PKL_REG), total data length (TOE_TDL_REG), and command for sending data (TOE_CMD_REG=Send).
5) Prepare file information and write to TOE_TXDATA_REG. The file information is sent to data connection through TOE25G-IP.
6) Wait until finishing data transmission by monitoring busy flag of TOE25G-IP (TOE_CMD_REG[0]='0').
7) Write command register of TOE25G-IP to close connection (TOE_CMD_REG=Close).
8) Wait until TOE25G-IP completes terminating the connection by monitoring busy flag of TOE25G-IP (TOE_CMD_REG[0]='0').
9) Call function to return FTP response 226 to TxRAM and start packet transmission.

3.2.2 STOR

1) Convert file name received from the client which is ASCII code to unsigned integer.
2) Set exFAT-IP parameters to store received data from FTP client to the SSD, i.e., file name (USRFNAME_REG) and command (EXFATCMD_REG=Write file). The number of files is fixed to 1 to transfer one file data at a time.
3) Wait until data connection establishment completes by monitoring ConnOn status (TOE_STS_REG[1]='1').
4) Call function to return FTP response 125 to TxRAM and start packet transmission.
5) Wait until data transmission is completed by monitoring busy flag of exFATNVMe (FATSTS_REG='0').
6) Wait until data connection is completely terminated by monitoring ConnOn status (TOE_STS_REG[1]='0').
7) Call function to return FTP response 226 to TxRAM and start packet transmission.

3.2.3 RETR

1) Convert file name received from the client which is ASCII code to unsigned integer.
2) Set exFAT-IP parameters to return data from the SSD to FTP client, i.e., file name (USRFNAME_REG) and command (EXFATCMD_REG=Read file). The number of files is fixed to 1 to transfer one file data at a time.
3) Call function to return FTP response 125 to TxRAM and start packet transmission.
4) Set TOE25G-IP parameters to send data from the SSD, i.e., transfer packet size (TOE_PKL_REG), total data length (TOE_TDL_REG), and command for sending data (TOE_CMD_REG=Send).
5) Wait until data transmission is completed by monitoring busy flag of TOE25G-IP (TOE_CMD_REG[0]='0'). The maximum file size of the SSD is 512 GB which is more than the maximum transfer size of TOE25G-IP (4 GB). So, step4) – 5) must be repeated many times for transferring more than 4 GB file size. Finally, total size of TOE25G-IP is equal to total size of exFATNVMe.
6) Follow step 7) – step 9) of LIST command to close the connection and return FTP response.

3.3 Function List in Test Firmware

The function in the firmware is split into three groups: exFAT-IP function, TOE25G-IP function and FTP function. The description of each function is shown as follows.

3.3.1 Function for exFAT-IP

Function in this topic is applied to control exFAT-IP.

void change_ftime(void)	
Parameters	None
Return value	None
Description	Print current create time and date by calling show_ftime function. After that, ask user to change the value. If input is valid, DATETIME_REG and global parameter (DateTime) are updated.

unsigned long long get_cursize(void)	
Parameters	None
Return value	Read value of FATCURTRNSIZEH/L_REG
Description	Read FATCURTRNSIZEH/L_REG and return read value as function result.

int format_fat(void)	
Parameters	None
Return value	0: User cancels command or command is finished. -1: Receive invalid input or error is found.
Description	Run Format command by following step. 1) Call change_ftime to set created date and time. 2) Read disk capacity from DCAPH/L_REG and calculate supported file size. 3) Set file size to USRFSIZE_REG from user input. 4) Set EXFATCMD_REG=Format command and wait until finishing operation. 5) Display the disk information by calling show_diskinfo function.

void show_diskinfo(void)	
Parameters	None
Return value	None
Description	Print the current disk information from global parameters, i.e., file size (DFsizeB), maximum file in the disk (TotalFCap), and total file in the disk (DFnum).

void show_ftime(void)	
Parameters	None
Return value	None
Description	Print current created date and time from global parameter (DateTime)

void show_size(unsigned long long size_input)	
Parameters	Size in byte unit
Return value	None
Description	Print input value in MB, GB, or TB unit

void show_result(void)	
Parameters	None
Return value	None
Description	Print total size by calling get_cursize and show_size function. After that, calculate total time usage from global parameters (timer_val and timer_upper_val) and display in usec, msec, or sec unit. Finally, transfer performance is calculated and displayed on MB/s unit.

void show_faterror(void)	
Parameters	None
Return value	None
Description	Read FATERRTYPE_REG and decode the value. Print error type when the flag is found. The example of error type is timeout error, NVMe-IP error, and unsupported disk capacity.

void update_dfsize(void)	
Parameters	None
Return value	None
Description	Read maximum number of files (TOTALFCAP_REG) and current file size in the disk (DFSIZE_REG) from hardware. File size is decoded and converted to be byte unit. Finally, maximum number of files and file size are updated to global parameters (TotalFCap and DFsizeB).

void update_dfnum(void)	
Parameters	None
Return value	None
Description	Read total number of files in the disk from DFNUM_REG, and then update read value to global parameter (DFnum).

int shutdown_dev(void)	
Parameters	None
Return value	0: Shutdown success, -1: Shutdown cancelled
Description	Set shutdown command to SSD (EXFATCMD_REG[1:0]=”01”) to change the SSD to inactive state. After that, busy flag (FAT_STS_BUSY[0]) is monitored until it is de-asserted to ‘0’.

3.3.2 Function for TOE25G-IP

Function in this topic is applied to control TOE25G-IP.

void exec_port(unsigned int port_ctl, unsigned int mode_active)	
Parameters	port_ctl: 1-Open port, 0-Close port mode_active: 1-Active open/close, 0-Passive open/close
Return value	None
Description	For active mode, write TOE_CMD_REG to open or close connection. After that, call read_conon function to monitor connection status until it changes from ON to OFF or OFF to ON, depending on port_ctl mode.

void init_toe_param(void)	
Parameters	None
Return value	None
Description	1) Set network parameters to TOE25G -IP register from global parameters. After reset is de-asserted, it waits until TOE25G-IP busy flag is de-asserted to '0'. 2) Set the header value and header enable for packet filtering in UserRxMAC. 3) Set default value to FTP response parameter.

int input_param(void)	
Parameters	None
Return value	0: Valid input, -1: Invalid input
Description	Receive network parameters from user, i.e., mode, window threshold, FPGA MAC address, FPGA IP address, FPGA port number, Target IP address, and Target port number. When the input is valid, the parameters are updated. Otherwise, the value does not change. After receiving all parameters, the current value of each parameter is displayed.

unsigned int read_conon(void)	
Parameters	None
Return value	0: Connection is OFF, 1: Connection is ON.
Description	Read value from USER_CMD_CONNON register and return only bit2 value to show connection status.

void show_param(void)	
Parameters	None
Return value	None
Description	Print the current value of the network parameters setting to TOE25G-IP such as IP address, MAC address, and port number.

3.3.3 Function for ftp operation

unsigned int cal_checksum(unsigned int byte_len, unsigned char *buf)	
Parameters	bytel_len: the number of bytes input for calculating checksum buf: the packet to calculate checksum
Return value	Checksum value
Description	Calculate 16-bit checksum of the data in the buffer

unsigned int decode_fname(void)	
Parameters	None
Return value	The value output from file name
Description	Convert filename from received packet which is ASCII code to the value

int read_rxpacket(void)	
Parameters	None
Return value	0: Packet is received, -1: Disconnect command from the client is received
Description	Wait for new received packet from UserRxMAC or receiving the disconnect command. When the new packet is received, the packet is copied to the receive temporary buffer for processing.

void send_list(void)	
Parameters	None
Return value	None
Description	Return file list with the information such as modified date and time for LIST command as described in topic 3.2.1.

void send_resp(unsigned int flag, unsigned int ftp_response)	
Parameters	flag: TCP flag in the received packet ftp_response: FTP response message
Return value	None
Description	Send FTP response by UserTxMAC

void set_tx_csum(void)	
Parameters	None
Return value	None
Description	Assign checksum to Tx header

void set_tx_header(unsigned int flag, unsigned int data_len)	
Parameters	flag: TCP Flag for transmitted packet data_len: Total transmit length in byte unit
Return value	None
Description	Set packet header of the transmitted packet including Ethernet header, IP header, and TCP header

void take_ftp_resp(void)	
Parameters	None
Return value	None
Description	Operate FTP command in Table 3-1 and Table 3-2 by creating FTP response and controlling data connection as described in topic 3.2.

void wait_ethlink(void)	
Parameters	None
Return value	None
Description	Read EMAC_LINKUP_STS[0] and wait until ethernet is linked up

4 Revision History

Revision	Date	Description
1.0	15-Oct-20	Initial version release